Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики, информатики и технологий
Кафедра информатики, информационных технологий
и методики обучения информатике

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДЛЯ УХОДА ЗА РАСТЕНИЯМИ

Выпускная квалификационная работа
бакалавра по направлению подготовки
09.03.02 – Информационные системы и технологии

Исполнитель: студент группы ИСиТ-1601
ИМФИиТ
Чернявская Н.В.

Руководитель: к.п.н., доцент кафедры
ИИТиМОИ
Арбузов С.С.

Допустить к защите
«____»_____ 2020 г.

Зав. кафедрой _____
М.В. Лапенок

Руководитель ОПОП _____
Л.В. Сардак

Екатеринбург – 2020

Реферат

Чернявская Н.В. АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДЛЯ УХОДА ЗА РАСТЕНИЯМИ, выпускная квалификационная работа: 56 стр., рис. 27, табл. 1, библ. 55 назв., приложений 2.

Ключевые слова: АВТОМАТИЗИРОВАННАЯ СИСТЕМА, РАСТЕНИЯ, ИНФОРМАЦИОННАЯ СИСТЕМА, МИКРОКОНТРОЛЛЕР, ВЕБ-ИНТЕРФЕЙС.

Предмет разработки – автоматизированная система для ухода за растениями.

Цель работы – спроектировать и разработать автоматизированную систему для ухода за растениями.

В рамках работы был произведен анализ существующих решений по автоматизации ухода за растениями, а также проанализирован и отобран инструментарий для реализации и последующей эксплуатации автоматизированной системы по уходу за растениями.

Система реализована на микроконтроллере ESP 12-E с поддержкой протокола связи Wi-Fi, который выступает в роли веб-сервера. Устройство осуществляет полив растений, обеспечение светом и управление настройками системы для ухода за растениями посредством веб-интерфейса.

Составлена техническая и сопроводительная документация по применению автоматизированной системы.

Оглавление

Введение

Для людей, имеющих комнатные растения, процесс полива во время отпусков, командировок или при необходимости полива растений строго по времени, сильно усложняется поисками знакомых или соседей, которым можно доверить такое ответственное дело. В век современных технологий можно поставить вопрос о том, как можно автоматизировать данный процесс.

Автоматизированная система по уходу за растениями − это незаменимая вещь, как для хозяина, так и для растений. С ней владелец может не беспокоиться о том, что его комнатное растение завянет из-за недостатка влаги или освещения. Данные устройства легко настраиваются, а так же могут иметь разные варианты программной и аппаратной части.

Предмет разработки: автоматизированная система для ухода за растениями.

Цель работы – спроектировать и разработать автоматизированную систему для ухода за растениями

Задачи:

1. Проанализировать состояние проблемы и подходов к ее решению.
2. Произвести анализ и обосновать выбор технологий реализации и необходимых программных платформ.
3. В соответствии с техническим заданием провести разработку автоматизированной системы для ухода за растениями.
4. Подготовить техническую и сопроводительную документацию по применению автоматизированной системы для ухода за растениями.

Глава 1. Теоретические основы создания информационных и радиоэлектронных систем для полива растений

1.1 Проектирование и разработка информационных систем для радиоэлектронных устройств

Перед созданием любого проекта необходимо произвести анализ существующих способов проектирования и инструментов для создания разрабатываемой информационной системы.

Информационная система (ИС) – совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств [3837].

Проектирование информационных систем – это упорядоченная совокупность методологий и средств создания или модернизации информационных систем [19].

Организацию проектирования информационных систем принято разделять на 2 типа:

- каноническое проектирование отражает особенности технологии оригинального (индивидуального) процесса;
- типовое проектирование направлено на выполнение проектирования информационных систем с использованием типовых проектных решений. Типовое проектное решение – решение, пригодное к многократному использованию (тиражируемое проектное решение).

Каноническое проектирование отличает отражение ручной технологии проектирования, осуществление на уровне исполнителей, использование инструментария универсальной компьютерной поддержки. Применяется каноническое проектирование, главным образом, для локальных и относительно небольших информационных систем с минимальным использованием типовых решений. Адаптация проектных решений происходит только посредством перепрограммирования программных модулей.

Для проектирования автоматизированной системы для ухода за растениями выбран канонический метод разработки системы.

Организовывается каноническое проектирование с использованием каскадной модели жизненного цикла (см. Рис. 1). Это предполагает разделение процесса на следующие стадии и этапы [30]:

1. Предпроектная стадия направлена на предпроектный анализ и составление технического задания. То есть, формируются требования к информационным системам, разрабатывается её концепция, составляется технико-экономическое обоснование и разрабатывается техническое задание;

2. Проектная стадия предусматривает составление эскизного и технического проекта, разработку рабочей документации;

3. Послепроектная стадия даёт старт мероприятиям по внедрению информационных систем, обучению персонала, анализу результатов испытания. Частью этой стадии становится сопровождение информационных систем и устранение выявленных недостатков.

Рис. 1. Каскадная модель жизненного цикла информационной системы

В настоящее время существует большое количество инструментальных средств, применяемых для реализации проекта информационной системы от этапа анализа до создания программного кода. Можно выделить так называемые CASE (Computer-Aided Software Engineering − Автоматизированная

Разработка Программного Обеспечения) − средства верхнего уровня (upper CASE tools) и CASE-средства нижнего уровня (lower CASE tools) [1].

Средством, позволяющим объединить эти подходы, явился унифицированный язык объектно-ориентированного моделирования (Unified Modeling Language – UML) [3]. К преимуществам языка UML можно отнести разнообразные инструментальные средства, которые как поддерживают жизненный цикл информационной системы, так и позволяют настроить и отразить специфику деятельности разработчиков различных элементов проекта.

UML обладает следующими основными характеристиками:

- является языком визуального моделирования, который обеспечивает разработку репрезентативных моделей для организации взаимодействия заказчика и разработчика информационных систем, различных групп разработчиков информационных систем;
- содержит механизмы расширения и специализации базовых концепций языка.

Для моделирования различных этапов жизненного цикла информационных систем язык UML предлагает различные типы диаграмм.

При разработке концептуальной модели применяют диаграммы вариантов использования и диаграммы деятельности, модели бизнес-объектов, диаграммы последовательностей.

На этапе работы над логической моделью информационной системы описать требования к системе позволяют диаграммы вариантов использования, а при предварительном проектировании используют диаграммы классов, диаграммы состояний, диаграммы последовательностей.

Детальное проектирование при разработке физической модели выполняют с применением диаграмм классов, диаграмм развертывания, диаграмм компонентов.

Благодаря диаграммам UML можно разработать концепцию системы, устройства и программного обеспечения.

Радиоэлектронное устройство (РЭУ) − радиоэлектронное средство, представляющее собой совокупность функционально и конструктивно законченных сборочных единиц и используемое для решения технической задачи в соответствии с его назначением [6].

Радиоэлектронный функциональный узел (РЭФУ) – радиоэлектронное средство, представляющее собой функционально и конструктивно законченную сборочную единицу, выполняющее радиотехническую и/или электронные функции(ию) и не имеющее самостоятельного применения [6].

Функциональный узел состоит из пассивных и активных элементов, соединение которых образует электрическую схему. Электронные устройства эффективны за счет быстродействия, точности и чувствительности входящих в нее элементов, важнейшими из которых являются электронные приборы.

Компоненты электронной техники также делятся на активные и пассивные. Активные элементы способны усиливать, обрабатывать и преобразовывать сигналы. Пассивные − накапливать или расходовать энергию сигнала.

Основными компонентами электронной техники являются: резисторы, катушки индуктивности и дроссели, конденсаторы, трансформаторы, коммутационные устройства, электровакуумные приборы, источники тока, приборы отображения информации, полупроводниковые приборы, пьезоэлектрические приборы, акустические приборы, антенны, линии задержки, предохранители и разрядники, электродвигатели, лампы накаливания, элементы аналоговой техники, элементы цифровой техники, провода, кабели [24].

Согласно ГОСТ 2.103-2013 «Стадии разработки» [8] при создании РЭА устанавливают следующие стадии проектной разработки: техническое предложение, эскизный проект, технический проект и их этапы.

Проектирование и разработка нового изделия требует затрат времени и высокой квалификации, поэтому разработку РЭУ и их конструкций осуществляют с применением следующих работ [1]:

- научно-исследовательской работы (НИР);
- опытно-конструкторской работы (ОКР).

Научно-исследовательская разработка – это комплекс теоретических и экспериментальных исследований, проводимых с целью получения исходных данных, изыскания принципов и путей создания или модернизации продукции, если таковых не имеется или они недостаточны непосредственно для успешной разработки изделия. Научные исследования в зависимости от содержания и характера получаемых результатов подразделяются на фундаментальные, поисковые и прикладные.

Целью фундаментальных исследований является открытие новых явлений, закономерностей и принципов, которые могут быть использованы при создании новой техники или технологии.

Поисковые научные исследования направлены на изучение конкретных проблем, например возможностей создания новых материалов, техники, технологии, повышения производительности труда и качества продукции и т. п. Результаты поисковых работ оформляются в виде отчетов, технической документации, макетов, экспериментальных образцов изделий.

Прикладные научные исследования направлены на создание новых изделий либо на совершенствование существующих, а также на разработку способов их производства, на разработку средств механизации и автоматизации производства, систем и методов контроля за качеством продукции и т. д. Прикладные НИР являются одной из стадий жизненного цикла изделий [18, 20].

Порядок выполнения НИР регламентируется ГОСТ 15.101-98 «Порядок выполнения научно-исследовательских работ» [11]. Стандарт устанавливает общие требования к организации и выполнению научно-исследовательских

работ; порядок выполнения и приемки НИР; этапы выполнения НИР, правила их выполнения и приемки; порядок разработки, согласования и утверждения документов в процессе организации и выполнения НИР; порядок реализации результатов НИР.

Задача опытно-конструкторской работы заключается в создании комплекта конструкторской документации для производства изделия. Данный вид работ является продолжением НИР или является самостоятельным. Он заканчивается выпуском полного комплекта технической документации на изделие, изготовлением и испытанием его опытного образца (или опытной партии).

РЭУ имеют жизненный цикл (см. Рис. 2), представляющий собой совокупность взаимосвязанных процессов изменения состояния продукции при ее создании, использовании (эксплуатации) и ликвидации [7].

Рис. 2. Жизненный цикл изделия

Таким образом, при проектировании информационных систем и радиоэлектронных устройств необходимо провести теоретические

исследования и анализ возможных методов и средств для разработки. При помощи методов моделирования UML создать диаграммы, отражающие различные этапы жизненного цикла информационных систем, что значительно упростит проектирование, формирование концепции системы, разработку технического проекта, разработку сопроводительной документации, а так же внедрение информационной системы.

1.2 Методы и средства реализации автоматизированных систем

Существует два основных способа обеспечения достаточной влагой растения в период отъезда из дома:

- сохранение влаги, которую растение получило в момент последнего полива;
- обеспечение растение водой без помощи человека.

Первый способ будет хорошим вариантом, когда поездка занимает небольшое количество времени, не более двух недель и дома нет растений, которые чувствительны к излишней влаге или растений, которым нужен четкий график полива. Для этого необходимо обильно полить растение, так чтобы земляной ком пропитался водой. Если растение находится в пористом горшке, например в глиняном, то горшок желательно обернуть мхом, который так же необходимо обильно увлажнить. В пластмассовых горшках почву сверху можно укрыть слоем мха, керамзита или гидрогеля. Политые растения размещаются вдали от батарей отопления и теплых мест, ставятся на поддоны с водой, так чтоб нижняя часть горшка оказалась в воде. Если растение не переносит нахождение корней в воде, то его оставляют в блюдечках, а стекшую воду после полива, сливают. Так же для уменьшения испарения влаги можно поместить растение под купол.

Второй способ подойдет тем, кто уезжает на длительное время. Реализовать полив растений, без участия людей можно множеством способов, в

том числе сделать полив своими руками или приобрести уже готовые приспособления или устройства для полива. Например, своими руками можно сделать фитильный полив – для этого нужно из полоски ткани или бинта скрутить фитиль и один конец поместить в емкость с водой, а второй расположить на поверхности земли в горшке или наоборот снизу горшка, возле корней растения.

Можно сделать полив при помощи пластиковой бутылки, наполненной водой. Для этого наполненную бутылку размешают в углублении в горшке, а в крышке бутылки проделывают отверстия – чем больше отверстий, тем интенсивней полив. Еще один из вариантов полива – медицинская капельница. Для этого необходимо поместить емкость с водой выше уровня горшка с растением и разместить в земле трубку от капельницы, а с помощью колесика, которым оснащены медицинские капельницы, можно регулировать напор струи.

Успешный рост растений зависит не только от полива, но и от света, который помогает им полноценно развиваться. В природной среде одни растения отлично растут в затемненных местах, другим нужен постоянный солнечный свет. При выращивании растений в условиях квартиры ситуация аналогична. Лампа для цветов комнатных создает постоянное освещение и отлично помогает растениям, особенно в зимний период.

Свет − главное условие для процесса фотосинтеза, который является питанием для растений. Листва содержит красящий пигмент − хлорофилл, который поглощает из атмосферы углекислый газ и воду и под воздействием солнечного ультрафиолета трансформирует их в кислород и углеводы. Они нужны для правильного роста и развития.

При недостаточном количестве света процесс начинает двигаться в обратном направлении, и цветок слабеет и гибнет. Для обеспечения достаточного питания, развития и роста побегов необходимо иметь дополнительное освещение для комнатных растений.

В зимний период практически всем растениям не хватает природного освещения в связи с сокращением солнечного света. Взрослым растениям необходимо 12 часов естественного освещения, молодой рассаде — около 16 часов. Поэтому устройства освещения необходимо регулировать в зависимости от природного светового дня. Утром и вечером приборы освещения следует включать на три-четыре часа. Главное условие для нормального роста зелени − регулярная подсветка [16, 42].

Автоматизированных систем для освещения в продаже нет, но в магазинах можно найти уже готовые решения автоматического полива растений. Существуют, как и примитивные системы полива, такие как колбы, которые необходимо наполнить водой, перевернуть и воткнуть в землю и самоорошающиеся цветочные горшки, так и автоматические системы полива.

Автоматические системы полива подразделяются на системы с уже предустановленной программой и на системы с возможностью настройки времени и полива.

В системах с предустановленной программой обычно уже настроена частота полива, а количество воды для полива регулируется не на программном уровне, а на механическом.

Системы с возможностью настройки обычно создаются на базе таймеров или микроконтроллеров. Некоторые из них имеют систему индикации, состоящую из дисплея или световых или звуковых индикаторов, а благодаря кнопкам или поворотным ручкам можно настраивать длительность и частоту полива. Стоимость таких систем варьируется от 1000 до 10000 тысяч рублей на момент написания ВКР.

Недостатком таких систем является отсутствие какой-либо обратной связи, например информации о влажности почвы, чтобы можно было удаленно узнавать это значение, а не измерять ее тактильным способом или «на глаз», а так же возможности полива не по таймеру, а исходя из значений влажности почвы. Различные растения в разные периоды жизни и в разное время года

требуют неодинаковой влажности почвы. Например, в весенне-летний период при усиленном росте все растения нуждаются в обильном поливе. Осенью рост растений замедляется, некоторые из них переходят в состояние покоя и поэтому требуют меньшего полива и увлажнения. В зимний период многие комнатные растения прекращают рост, некоторые сбрасывают листья и требуют временного покоя. Такие растения поливают очень мало, но следят, чтобы земля в горшке не пересыхала, а всегда была влажной. Нормальное состояние горшечных растений во многом, зависит от правильного и своевременного их полива [44]. Системы полива по таймеру будут нуждаться в своевременной корректировке значений интервала и длительности поливов, иначе растения могут засохнуть или наоборот погибнуть от чрезмерной влажности.

Еще одним минусом таких устройств является невозможность настройки системы, находясь вдали от неё.

В наше время развитие компьютеров и мобильных устройств шагнуло далеко вперед, что позволило стать им общедоступными, и на сегодняшний день редко можно найти человека, который не имеет дома компьютер, смартфон или планшет. Так же стоит отметить, что Интернет развиваются с огромной скоростью. В своем отчете о состоянии интернет-среды – Digital 2020 [46] компании We Are Social и Hootsuite подмечают (см. Рис. 3), что:

- число людей во всем мире, пользующихся интернетом, выросло до 4,54 миллиарда, увеличившись на 7 процентов (298 миллионов новых пользователей) по сравнению с январем 2019 года;
- во всем мире в январе 2020 года насчитывается 3,80 миллиарда пользователей социальных сетей, причем это число увеличилось более чем на 9 процентов (321 миллион новых пользователей) с этого времени прошлого года;

- в настоящее время в мире мобильными телефонами пользуются более 5,19 миллиарда человек, а число пользователей увеличилось за последний год на 124 миллиона (2,4 процента).

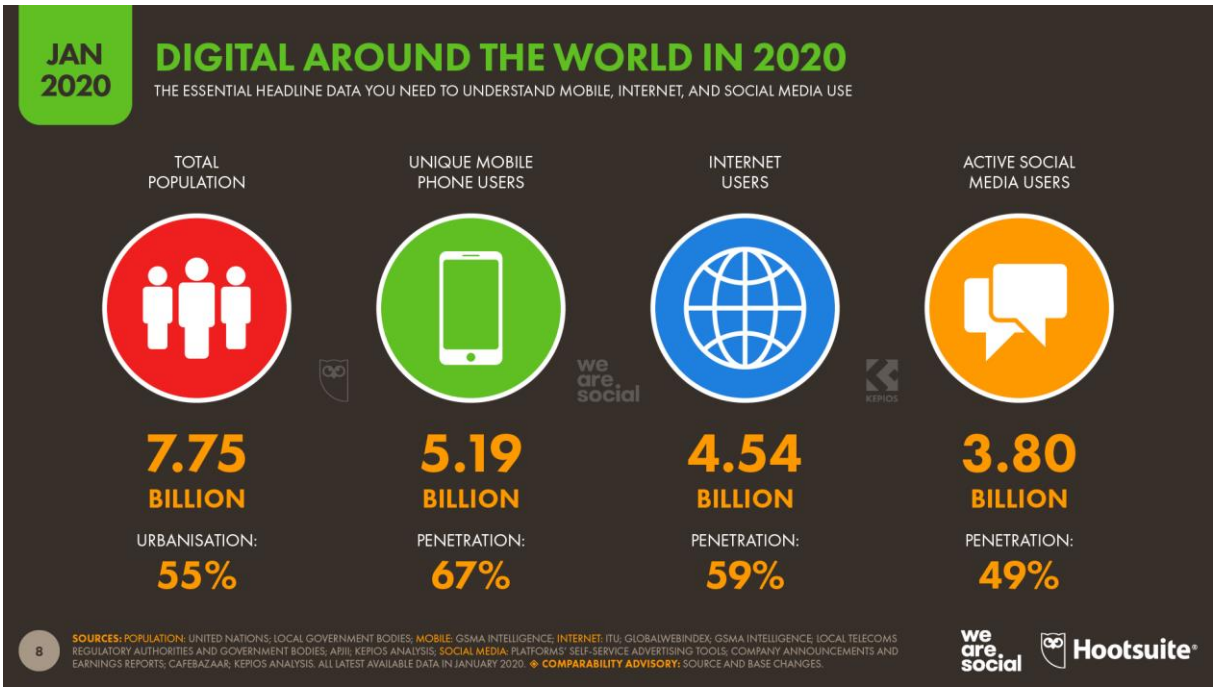

Рис. 3. Наиболее важные цифры глобального отчета Digital 2020[1]

Поэтому наиболее удобным и современным вариантом управления системой по уходу за растениями будет являться управление средствами веб-браузера с персонального компьютера или мобильного устройства.

Любое устройство имеет свое строение и принцип работы, но оно должно включать в себя контроллер, для управления всеми компонентами устройства. В настоящее время роль таких контроллеров чаще всего выполняет микроконтроллер. Микроконтроллер − это микросхема, предназначенная для управления электронными устройствами. Базовое исполнение микроконтроллера [40] включает в себя один полупроводниковый кристалл и состоит из центрального процессора, в который включены блоки управления, регистры, постоянное запоминающее устройство (ПЗУ) и периферия, которая

[1] https://wearesocial-net.s3.amazonaws.com/uk/wp-content/uploads/sites/2/2020/01/01-Global-Overview-–-DataReportal-Digital-2020-Global-Digital-Overview-Slide-8.png

состоит из портов ввода-вывода, контроллеров прерываний, таймера, генератора импульсов, аналоговых преобразователей и других элементов.

Микроконтроллеры имеют различные характеристики: архитектура ядра, размер и типы памяти, максимальная частота работы, количество линий ввода-вывода, наличие тех или иных периферийных устройств, напряжение питания и другие. Существуют следующие известные семейства микроконтроллеров:

- MSP430 (Texas Instruments) − имеют фоннеймановскую архитектуру и содержат 16-битное RISC ЦПУ, периферийные модули, а также гибкую систему тактирования, объединённые общими шинами адреса и данных. Сочетание современного ЦПУ и отображаемых в памяти аналоговых и цифровых периферийных модулей делает семейство MSP430 пригодным для работы в приложениях, связанных с обработкой смешанных сигналов [35];
- MCS 51 (Intel) − монокристальный МК на основе Гарвардской архитектуры. Основной представитель этого семейства 80C51, созданный по технологии CMOS. И хотя эти контроллеры разработаны еще в 80-х годах прошлого века, но до сих пор широко применяются. И сегодня многие компании, такие как Siemens, Philips и др. выпускают свои контроллеры с подобной архитектурой;
- STM (STMicroelectronics) − 8-разрядные контроллеры (STM8) относятся к категории высоконадежных с низким энергопотреблением изделий. В это же семейство входят контроллеры STM32F4 и STM Их основу составляет-32 битный Cortex. Такие контроллеры обладают отлично сбалансированной архитектурой и имеют большие перспективы развития.
- ARM (ARM Limited) − разработаны на собственной архитектуре. В семейство входят 32-х и 64-битовые МК. ARM Limited занимается только разработкой ядер и их инструментов, а лицензии на производство продает другим компаниям. Эти процессоры потребляют мало энергии, поэтому находят широкое применение в производстве мобильных телефонов,

игровых консолей, маршрутизаторов и т. д. К компаниям, выкупившим лицензии, относятся: STMicroelectronics, Samsung, Sony Ericsson и др.;

- AVR (Atmel) (ATmega, ATtiny, XMega) − Высокоскоростные контроллеры разработаны на собственной архитектуре. Основой контроллера является Гарвардский RISC-процессор с самостоятельным доступом к памяти программ и баз данных (Flash ПЗУ). Каждый из 32 регистров общего назначения может работать как регистр-аккумулятор и совокупность 16-битных команд. Высокая производительность в 1 MIPS на каждый МГц тактовой частоты обеспечивается за счет порядка выполнения команд, который предусматривает выполнение одной команды и одновременную подготовку к следующей. Для поддержания своей продукции компания Atmel выпускает бесплатную и качественную среду разработки Atmel;
- PIC (Microchip) − МК Гарвардской архитектуры. В его основе лежит архитектура с сокращенным набором команд, встроенная память команд и данных, низкое энергопотребление. В это семейство входят более 500 различных МК (8-ми, 16-ти, 32-битные) с различными наборами периферии, памяти и прочими характеристиками [22];
- ESP8266 и ESP32 (Espressif) – новое семейство микроконтроллеров представляет собой систему, под управлением 32-битного процессора Tensilica Xtensa на одном кристалле с интегрированным Wi-Fi и Bluetooth (ESP32) или с интегрированным Wi-Fi (ESP8266), который имеет возможность исполнять программы из флеш-памяти. Важной особенностью является отсутствие пользовательской энергонезависимой памяти на кристалле [55].

Для создания веб-сервера и веб-интерфейса необходим микроконтроллер поддерживающий протокол связи Wi-Fi и предоставляющий HTTP-интерфейс внешнего доступа. Такие микроконтроллеры представлены только у компании Espressif, а именно микроконтроллеры ESP8266 и ESP32, но так как при

разработке устройства протокол передачи данных Bluetooth не является необходимым, был выбран микроконтроллер ESP8266.

Существует около двадцати различных версий модулей, использующих данный микроконтроллер, различающиеся количеством выведенных контактов, схемой расположения выводов (распиновкой), габаритами печатной платы, объемом Flash-памяти и типом встроенной антенны или возможностью подключения внешней антенны. Были сравнены три распространённых версий модулей на базе микроконтроллера ESP8266 (см. Таблица 1).

Таблица 1

Сравнение версий модулей на базе ESP8266

Характеристики	Версии модулей		
	ESP-01	ESP-07	ESP-12E
Порты ввода/вывода	8	16	22
Порты ввода/вывода свободного назначения	4	9+1 (АЦП)	17+1 (АЦП)
Антенна	PCB (дорожка на плате)	Встроенная керамическая или коннектор U.Fl	PCB (дорожка на плате)
Частота процессора	80 МГц	80 МГц	80 и 160 МГц
Flash-память	1 Мб	1 Мб	4 МБ
Объём оперативной памяти	96 Кб	80 Кб – данных 32 Кб - инструкций	36 Кбайт
Размеры	14,3 x 24,8 x 3 мм	16 x 21,2 x 3 мм	16 x 24 x 3мм

Для устройства необходим вывод с поддержкой аналого-цифрового преобразования для подключения датчика влажности, а так же необходим большой размер Flash-память для создания и хранения веб-интерфейса. Таким

образом, для разработки автоматизированной системы ухода за растениями был выбран микроконтроллер ESP8266 версии ESP-12E.

Микроконтроллер ESP-12E [47] однокристальная система, построенная на базе процессора Tensilica L106 с ультранизким энергопотреблением, разрядность 32-бит. Поддерживает тактовые частоты 80 и 160 МГц. Поддерживает стандарты IEEE802.11 b/g/n, полный стек TCP/IP протоколов, три режима работы Wi-Fi: станция, программная точка доступа, станция + программная точка доступа (STA, softAP, STA+softAP). Имеет 17 выводов общего назначения, в том числе выводы с поддержкой аналого-цифрового преобразования (АЦП) и широтно-импульсной модуляции (ШИМ), так же включены интерфейсы UART, SPI, HSPI, SDIO, I2C, I2S, IrDA. На плате имеется антенна, разведенная в виде дорожки и встроенная Flash-память 4 Мбайт.

Данный микроконтроллер имеет различные варианты прошивок со встроенными интерпретаторами разнообразных языков высокого уровня:

- C − стандартизированный процедурный язык программирования, является одним из стандартных языком для программирования микроконтроллеров. Его плюсы: поддерживается многими средами разработки, имеет множество библиотек для работы с аппаратной частью устройства, прост в освоении, является оптимальным по скорости разработки и оптимизации среди языков;
- JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. При использовании микроконтроллера с возможностью создания своего веб-сервера и веб-страницы язык JavaScript будет удобно использовать для реализации этого, но для работы с аппаратной частью данного языка может быть недостаточно;

- Python − высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика, читаемости кода и на разработку веб-приложений. Синтаксис ядра Python минималистичен. Код в Python организовывается в функции и классы, которые могут объединяться в модули;
- MicroPython − это реализация языка программирования Python 3, которая включает основную часть подмножества стандартной библиотеки Python и оптимизирована для работы на микроконтроллерах;
- Lua − скриптовый язык программирования, по своей идеологии и реализации язык Lua ближе всего к языку программирования JavaScript, в частности, он также реализует прототипную модель объектно-ориентированного программирования, но отличается Паскале-подобным синтаксисом и более мощными и гибкими конструкциями.

Из приведенных выше языков программирования микроконтроллеров наиболее оптимален язык С, так как для него существует множество специализированных библиотек для работы с электронными компонентами аппаратной части комплекса.

Для микроконтроллера ESP-12E существуют свои программные средства разработки (программный комплект разработчика, SDK) состоящие из:

- компилятора. Компилятор для процессора Xtensa LX106 входит в пакет компиляторов GNU Compiler Collection. Он имеет открытые исходные тексты. В разных SDK могут содержаться разные сборки этого компилятора, немного отличающиеся поддерживаемыми опциями;
- библиотек для работы с периферией контроллера, стеков протоколов Wi-Fi, TCP/IP;
- средств загрузки исполняемого файла в память программ микроконтроллера;
- опциональной IDE (Integrated Development Environment − интегрированная среда разработки).

Компания разработчик Espressif имеет свой комплект разработчика и свободно его распространяет. В этот комплект входит компилятор GCC, библиотеки Espressif и загрузочная утилита XTCOM [52]. Библиотеки поставляются в виде скомпилированных библиотек, без исходных текстов.

Существует ряд альтернативных проектов и дополнений к уже существующим средам разработки помимо официальной SDK. Сторонние SDK используют библиотеки Espressif или предлагают собственный эквивалент данных библиотек, полученный методами реверсинжиниринга:

- Unofficial Development Kit [31]. В комплект входит Windows-инсталлятор, компилятор GCC собственной сборки с интеграцией с графической IDE Eclipse, актуальные комплекты библиотек и документации Espressif, а также некоторые дополнительные утилиты;
- Arduino IDE for ESP8266 [25] − дополнение к IDE Arduino, позволяющее программировать ESP8266 так же легко как любые другие модули Arduino. При этом доступна сетевая функциональность ESP8266. Включает в себя компилятор GCC, загрузчик прошивки ESPTool;
- PlatformIO [54] – расширение для интегрированной среды разработки Visual Studio Code, позволяющее легко интегрировать специальные

функции разработки устройств с поддержкой беспроводной связи, такие как удаленное обновление прошивки и тестирование.

Из приведенных выше сред разработки программ для микроконтроллера ESP-12E наиболее оптимальна Visual Studio Code с расширением PlatformIO, она поддерживает все микроконтроллеры, содержит компилятор GNU C/C++ и эмулятор, позволяющий отладить выполнение программы без загрузки в микроконтроллер, предлагает множество расширенных функций, таких как автозаполнение и Intellisense (автодополнение).

Одной из важнейших функций, которую обеспечивает микроконтроллер ESP-12E, является способность не только подключаться к существующей Wi-Fi сети и работать как веб-сервер, но также создавать собственную сеть, позволяя другим устройствам подключаться к нему и получать доступ к веб-страницам [36]. Управление, основанное на беспроводной связи с микроконтроллером, обеспечивается посредством веб-интерфейса, работающим на веб-сервере, который создает сам микроконтроллер. Таким образом, через сервер обеспечивается управление системой по локальной сети, как с домашнего персонального компьютера, так и с мобильных устройств посредством Wi-Fi технологии.

Взаимодействие в глобальной сети Интернет разных приложений, устройств и узлов происходит благодаря технологиям веб-служб (веб-серверов), основанных на определенных протоколах и соглашениях.

На сегодняшний день наибольшее распространение получили следующие протоколы реализации веб-сервисов:

- SOAP (Simple Object Access Protocol) – протокол обмена структурированными сообщениями в распределенной вычислительной среде. Также предоставляет стандарт структуры упаковки данных для транспортировки XML документов с помощью различных интернет-технологий, как: SMTP, HTTP, FTP [4]. SOAP протокол обычно

используется совместно со стандартами WSDL/UDDI, образующие так называемый «треугольник СОА» (см. Рис. 4).

Рис. 4. Процесс взаимодействия между клиентом и поставщиком веб-сервиса («треугольник СОА»)

- REST (Representational State Transfer) − это стиль архитектуры программного обеспечения для распределенных систем, использующий веб-протоколы и веб-технологии. REST архитектура включает в себя клиентское и серверное взаимодействие, построенное вокруг передачи ресурсов [50]. Веб-сервера, реализуемые с использованием протоколов HTTP и принципов REST, называют «веб-сервис RESTful» [17].

Выбор технологии необходимо рассматривать в контексте реализуемой системы и ее ограничений, но SOAP и REST можно сравнить по основным параметрам [32, 41, 51]:

1. Формат обмена сообщениями.
   1.1. SOAP используется формат XML для запросов и ответов;
   1.2. В REST нет фиксированного формата, возможен обмен сообщениями на основе XML, JSON или любого другого удобного формата. Передача данных в формате JSON осуществляется быстрее и обрабатывается легче, чем в формате XML. Также в большинстве современных языков программирования присутствует поддержка этого формата;

2. Время обработки запросов:
    2.1. SOAP – это целое семейство протоколов и стандартов. Из этого следует, что это более тяжеловесный и сложный вариант с точки зрения машинной обработки;
    2.2. REST не использует конвертацию данных при передаче, данные передаются в исходном виде − это снижает нагрузку на клиент веб-сервиса, но увеличивает нагрузку на сеть;
3. Транспортные протоколы:
    3.1. SOAP не накладывает никаких ограничений на тип транспортного протокола. Вы можете использовать либо веб-протокол HTTP, либо MQ;
    3.2. REST базируется на транспортном протоколе HTTP. Это означает, что все существующие наработки на базе протокола HTTP, такие как кеширование на уровне сервера, масштабирование, продолжают работать в REST архитектуре;
4. Простота реализации:
    4.1. При работе с протоколом SOAP необходимо определить свой сервис с использованием WSDL, так же обработка XML-данных сложный и ресурсоемкий процесс;
    4.2. REST обычно использует JSON, преимущества, которого описаны выше. В дополнение к этому, REST не требует наличия определения службы для предоставления веб-службы.

Обычно, SOAP используется в крупных корпоративных системах со сложной логикой, когда требуются четкие стандарты, подкрепленные временем. При разработке публичных API (Application Programming Interface − программный интерфейс приложения), где логика взаимодействия во многом покрывается четверкой методов CRUD (create, read, update, delete) – выбирают архитектуру REST. Он наиболее популярен в сети Интернет. Например, Яндекс, Google и другие используют именно его для своего API [33].

Для реализации веб-интерфейса используется языки HTML, CSS и JavaScript:

- HTML − язык гипертекстовой разметки, с помощью которого создаётся структура страницы;
- CSS − каскадная таблица стилей. С помощью этого языка задаются правила стилизации компонентов на странице;
- JavaScript − это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах.

Таким образом автоматизированная система для ухода за растениями будет реализована на микроконтроллере ESP-12E с программным обеспечением на языке C, созданным при помощи SDK Visual Studio Code с расширением PlatformIO. Управление системой будет осуществляться посредством веб-интерфейса, реализованным согласно принципам REST.

1.3 Формализованное описание технического задания

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на разработку автоматизированной системы для ухода за растениями

Составлен на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» [15].

1. Общие сведения.
   1.1. Название организации-заказчика.
   ФГБОУ ВО «УрГПУ».
   1.2. Название продукта разработки (проектирования).
   Автоматизированная система для ухода за растениями.
   1.3. Назначение продукта.
   Осуществление полива растений, обеспечение светом и управление настройками системы для ухода за растениями посредством веб-интерфейса.

1.4. Плановые сроки начала и окончания работ.

В соответствии с планом выполнения ВКР (01.09.2019 – 19.05.2020).

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки.

Автоматизированная система для ухода за растениями предназначена для домашнего использования.

2.2. Характеристика персонала (количество, квалификация, степень готовности)

С продуктом работает пользователь с базовыми знаниями о работе с ПК и веб-браузерами.

3. Требования к продукту разработки.

3.1. Требования к продукту в целом.

Данная система предназначена для домашнего использования и управлением системой посредством персонального компьютер или мобильного устройства с использованием браузера.

3.2. Аппаратные требования.

Для функционирования данной системы необходим персональный компьютер или смартфон с возможностью подключения к сети Интернет, через роутер.

3.3. Указание системного программного обеспечения (операционные системы, браузеры, программные платформы и т.п.).

Для данной системы необходим любой веб-браузер.

3.4. Указание программного обеспечения, используемого для реализации.

ПО для разработки

Для разработки системы необходимо следующее ПО:

- Редактор исходного кода − Visual Studio Code с дополнением PlatformIO;
- Текстовый редактор с возможностью подсветки синтаксиса языков программирования − Notepad++;
- Веб-браузер с поддержкой браузерных инструментов разработчика.

3.5. Форматы входных и выходных данных

Входные данные должны формироваться аппаратной частью системы, а именно датчиками.

3.6. Источники данных и порядок их ввода в систему (программу), порядок вывода, хранения.

Данные должны формироваться аппаратной частью системы, а так же могут вводиться с клавиатуры пользователем, через веб-интерфейс.

3.7. Меры защиты информации.

Не предусмотрено.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

Интерфейс должен быть интуитивно понятным и комфортным для пользователя системы, основной функционал реализован в рамках одной страницы (см. Рис. 5). Авторизация в сети предусматривается на отдельной странице (Рис. 6).

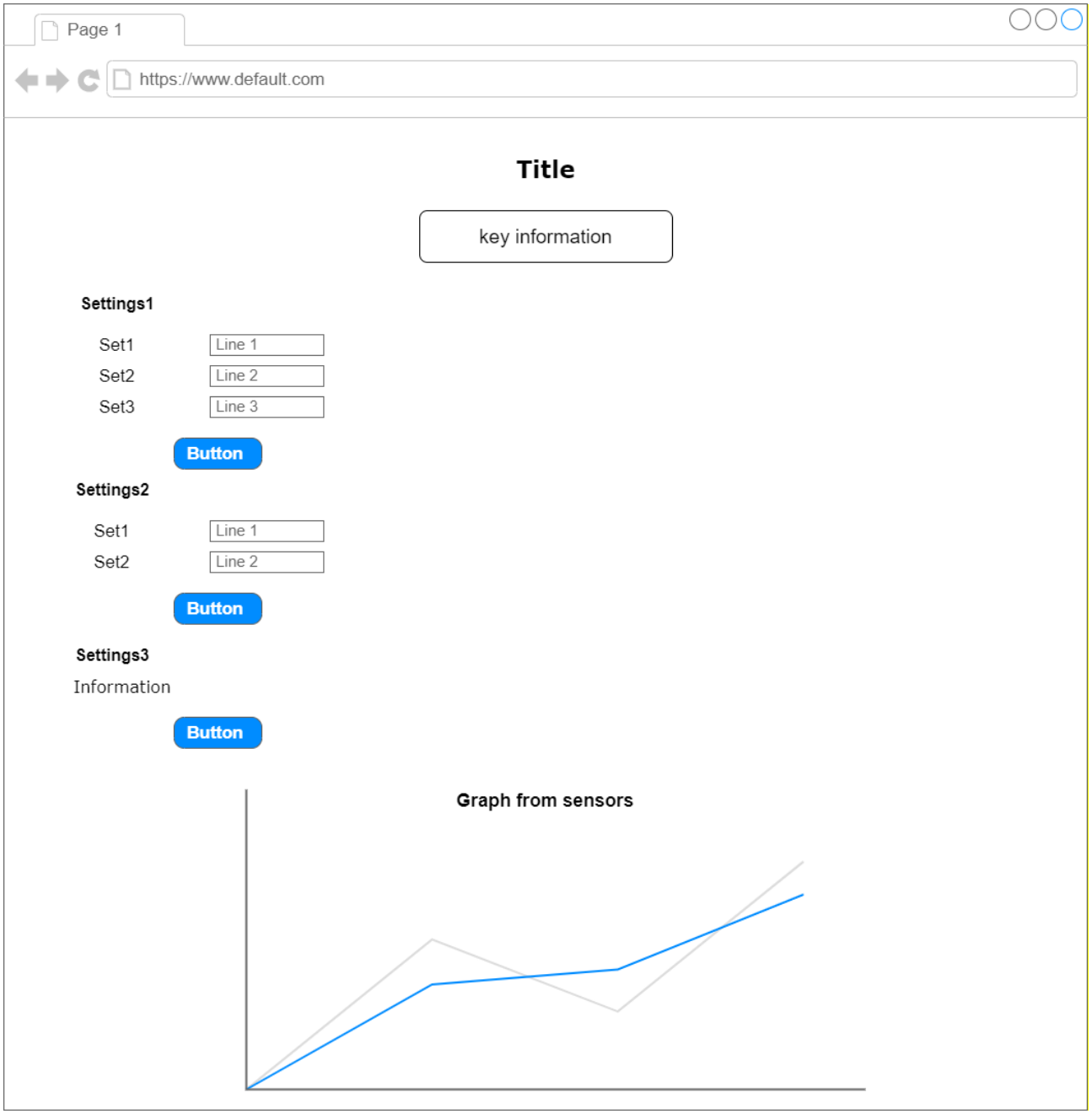

Рис. 5. Веб-интерфейс системы

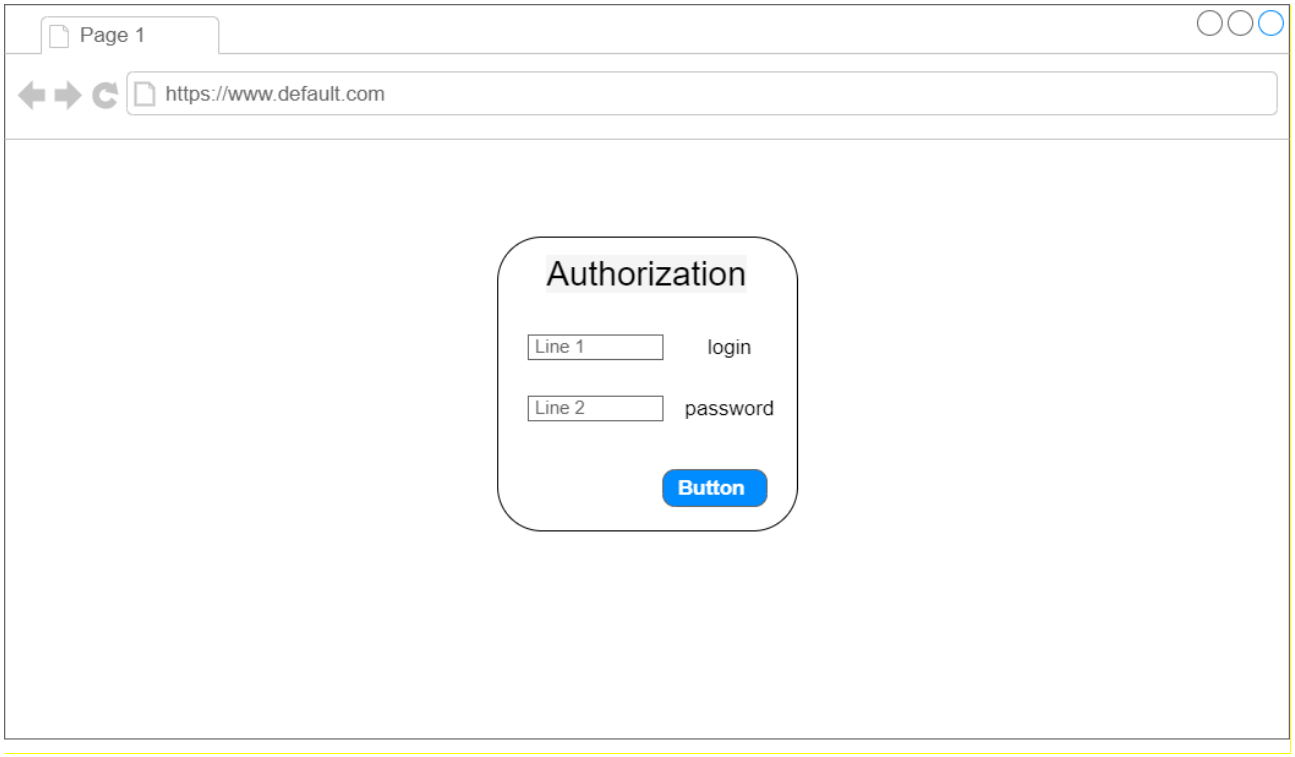

Рис. 6. Веб-интерфейс авторизации

4.2. Особенности ввода информации пользователем, представление выходных данных.

Система должна осуществлять проверку корректности введенных данных.

5. Требования к документированию.

5.1. Перечень сопроводительной документации.

Руководство пользователя автоматизированной системой для ухода за растениями.

5.2. Требования к содержанию отдельных документов.

Не предусмотрено

6. Порядок сдачи-приемки продукта.

В соответствии с планом выполнения ВКР.

Глава 2. Разработка автоматизированной системы ухода за растениями

2.1 Модельные представления объекта разработки

Перед написанием программного обеспечения и созданием аппаратной части устройства необходимо получить сведения о структуре, описать функциональные требования к системе, решить каким образом система будет взаимодействовать с пользователем.

Описание системы с помощью IDEF0 (Icam DEFinition) называется функциональной моделью. Основу методологии IDEF0 составляет графический язык описания процессов.

Методология IDEF0 предписывает построение иерархической системы диаграмм − единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция − система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности [21].

В процессе проектирования была разработана функциональная модель (см. Рис. 7) и её декомпозиция (см. Рис. 8), которая описывает основные части работы системы и их взаимодействие между собой.

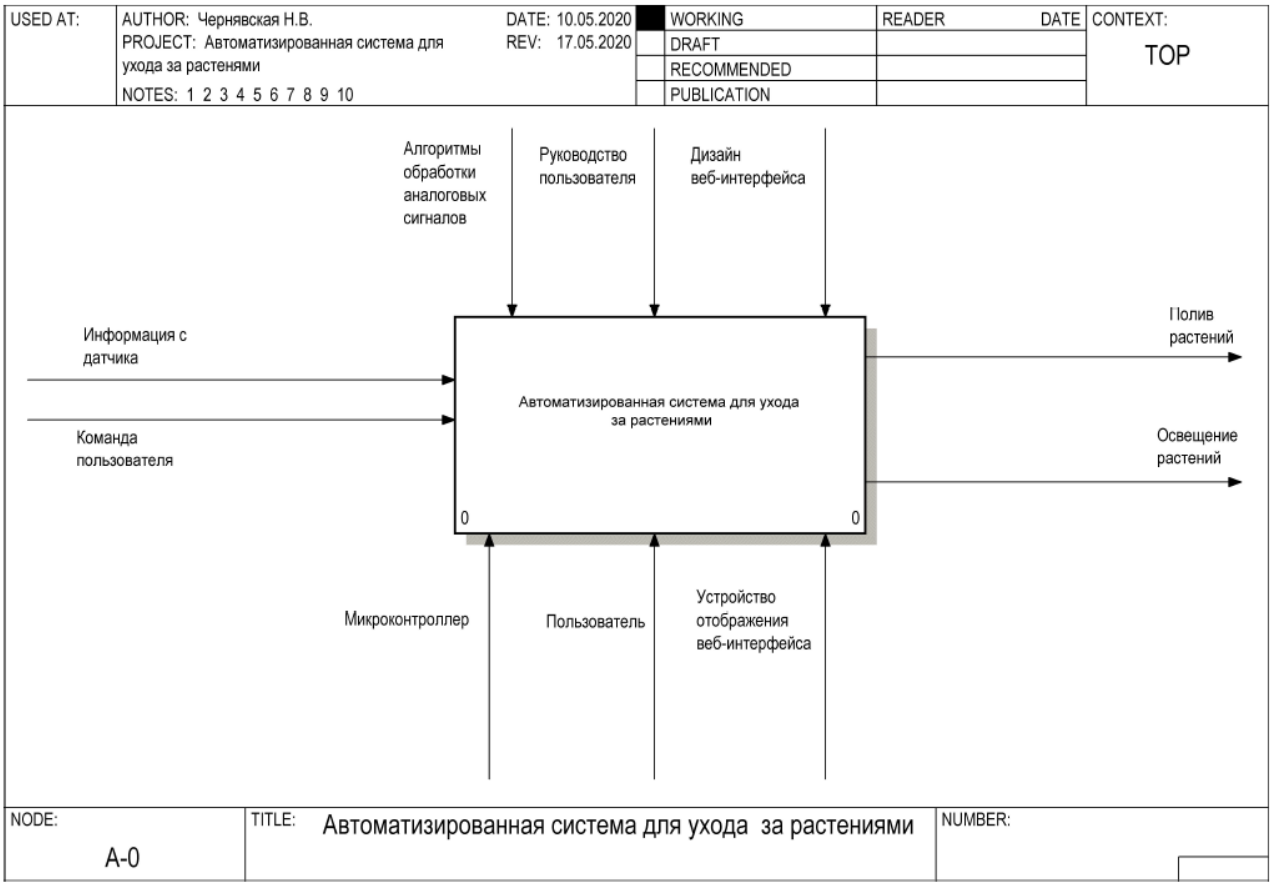

Рис. 7. Функциональная модель «Автоматизированной системы для ухода за растениями»

Входными параметрами системы, подающимися слева, являются команды пользователя, вводимые посредством веб-интерфейса и информация с датчика влажности почвы. Выходом является результат выполнения программы – полив или освещение растений. Управляющими элементами являются:

- алгоритм обработки аналоговых сигналов, благодаря которому происходит обработка данных с датчика;
- руководство пользователя, которое помогает пользователю взаимодействовать и настраивать систему;
- дизайн веб-интерфейса.

Механизмами системы являются микроконтроллер, пользователь, устройство отображения веб-интерфейса (персональный компьютер, смартфон, планшет).

Микроконтроллер участвует во всех операциях:

- обработки сигнала, считываемого с датчика;

- отображении веб-интерфейса;
- обработки команд, которые отправил пользователь.

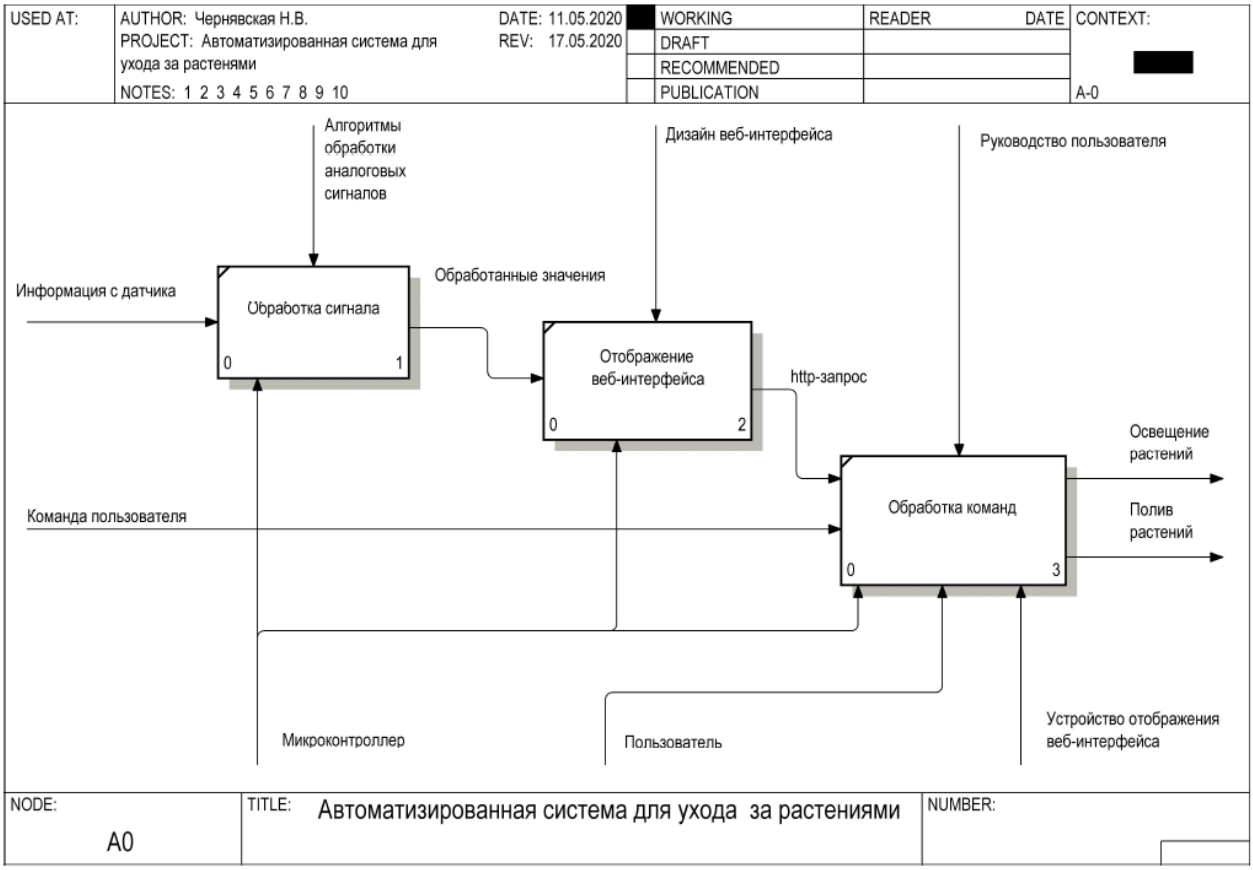

Рис. 8. Декомпозиция функциональной модели «Автоматизированной системы для ухода за растениями»

Для наглядного описания взаимодействия пользователям с системой был использован унифицированный язык моделирования UML.

Моделирование информационных систем при помощи UML начинают с диаграммы вариантов использования (use case diagram), поскольку она позволяет описать возможные варианты взаимодействия с системой (см. Рис. 9).

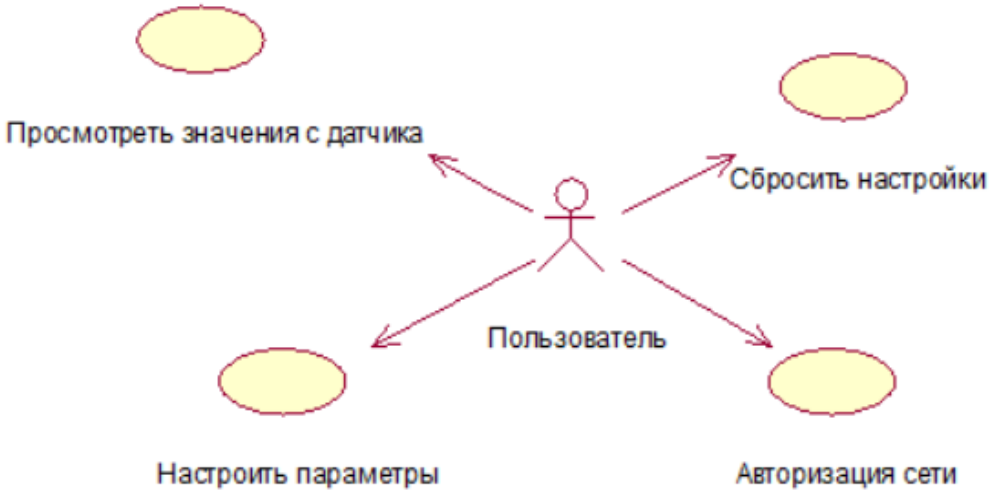

Рис. 9. Диаграмма вариантов использования

Разработанная диаграмма последовательности (sequence diagram) отображает взаимодействие объектов при выполнении полива растений (см. Рис. 10). Таким же образом может выглядеть диаграмма последовательности для применения освещения растений, только без использования датчика.

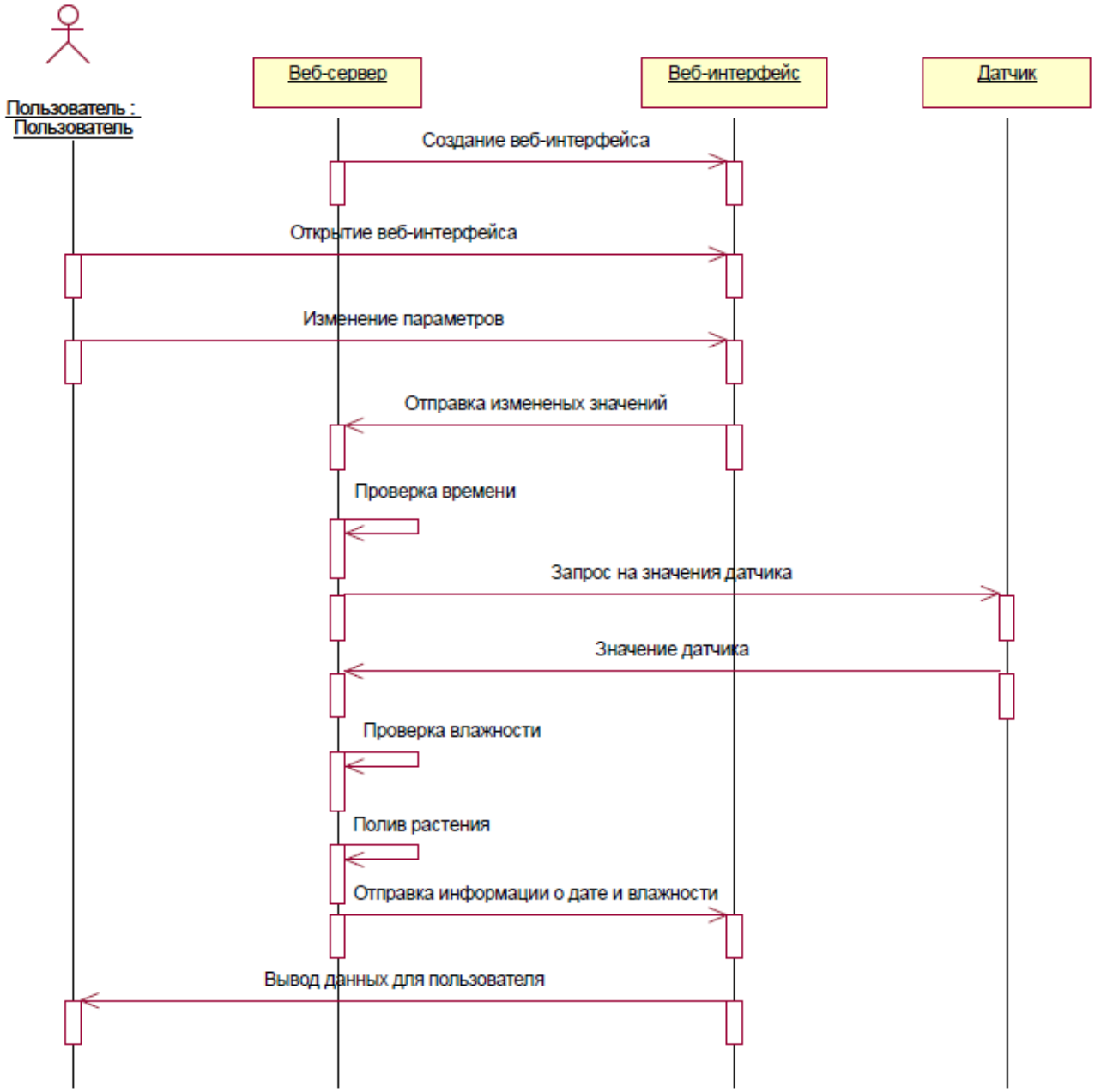

Рис. 10. Диаграмма последовательности

Диаграммы деятельности (activity diagrams) можно использовать для моделирования динамических аспектов поведения системы. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения. Были созданы диаграммы, показывающие основные сценарии, которые может выполнять пользователь:

1. Авторизация в домашней сети Wi-Fi (см. Рис. 11);

Рис. 11. Диаграмма деятельности: авторизация в сети Wi-Fi

2. Просмотр и настройка параметров полива, а именно изменение частоты измерения датчика, и параметры порогов максимальной и минимальной влажности (см. Рис. 12);

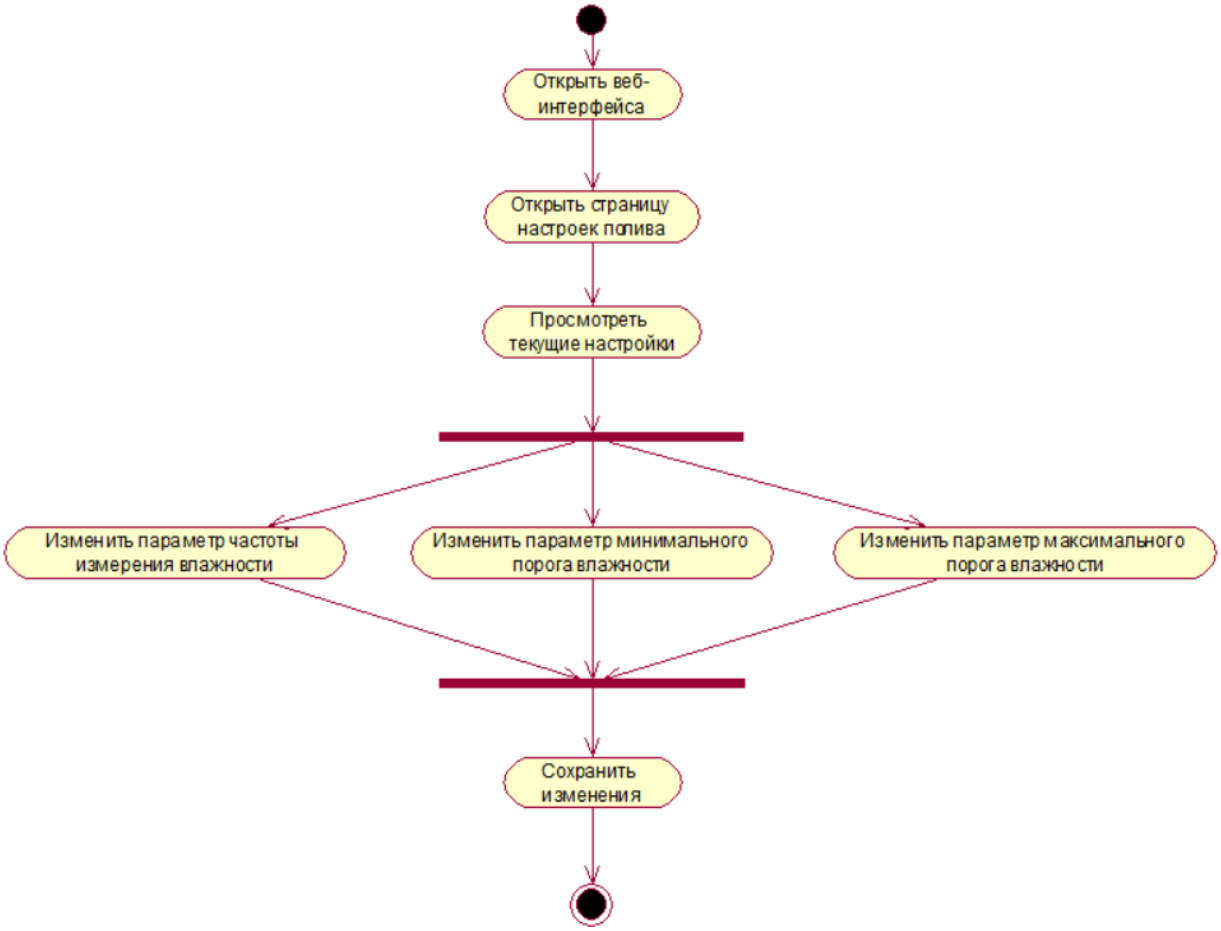

Рис. 12. Диаграмма деятельности: настройка полива

3. Просмотр и настройка параметров освещения, а именно изменение времени начала и окончания работы лампы (см. Рис. 13);

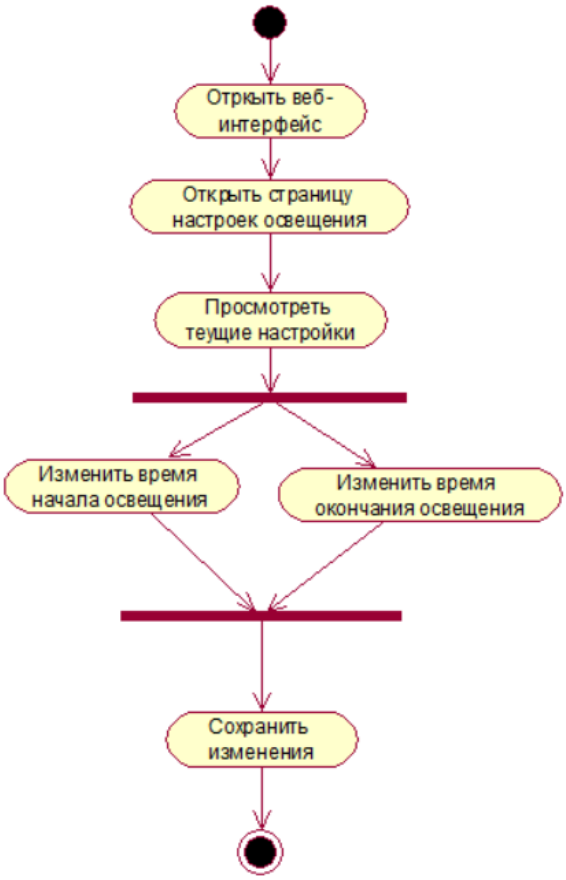

Рис. 13. Диаграмма деятельности: настройка освещения

4. Сброс текущих настроек полива, освещения и авторизации (см. Рис. 14).

Рис. 14. Диаграмма деятельности: сброс параметров

Таким образом, создание IDEF0 и UML диаграмм значительно упрощает и делает более понятной и структурированной разработку программной и аппаратной части устройства ухода за растениями.

2.2 Описание аппаратной и программной части автоматизированной системы

Для создания автоматизированной системы по уходу за растениями необходимо реализовать аппаратную и программную части системы.

Аппаратная часть реализована на микроконтроллере ESP-12E, который используется как веб-сервер для создания веб-интерфейса и контроллер для реализации полива и освещения растений. Полив растений происходит на основании настроек пользователя и данных, подающихся с датчика влажности почвы. Вода для полива подается при помощи погружного насоса, под управлением реле.

Написание программного обеспечения для устройства можно разделить на две части: создание веб-интерфейса (Приложение 2) и написание программы для микроконтроллера (Приложение 1).

Программа для микроконтроллера ESP12-E написана на языке C с подключением следующих библиотек с лицензией GNU Lesser GPL, которая позволяет использовать их как в свободном программном обеспечении, так и в коммерческом:

- EEPROM.h – используемая для работы с энергонезависимой памятью EEPROM, в которой будут храниться данные для авторизации пользователя в сети Wi-Fi;
- time.h, NTPClient.h – применятся для получения времени и даты по протоколу NTP, синхронизации времени и работе с ним;
- WiFiUdp.h − необходима для UDP-связи между микроконтроллером и внешними клиентами;
- FS.h – библиотека для работы с SPIFFS (Serial Peripheral Interface Flash File System) [48] файловой системой флеш-памяти, подключаемой по последовательному периферийному интерфейсу;
- ESP8266WebServer.h – библиотека для создания веб-сервера, отправки сообщений на сервер;

- ESP8266FtpServer.h – позволяет создавать FTP-сервер и работать с ним, а так же записывать данные в SPIFFS-память (лицензия GNU Lesser GPL v.2.1).

А так же библиотеки:

- ESP8266WiFi.h – основанная библиотека для работы с Wi-Fi сетью, позволяет выбирать режим работы микроконтроллера, отвечает за подключение к локальным точкам доступа и к локальным серверам. Библиотека имеет лицензию MIT [53], то есть позволяет использовать лицензируемый код в закрытом программном обеспечении при условии, что текст лицензии предоставляется вместе с этим программным обеспечением.
- ArduinoJson.h – библиотека для работы с форматом JSON с поддержкой сериализации, десериализации, нулевого копирования, потоков и фильтрации. Библиотека имеет лицензию MIT.

При запуске устройства происходит инициализация переменных, конфигурация времени, настройка выводов микроконтроллера, к которым подключены реле и датчик, на вывод. Затем происходит чтение двух байт из EEPROM-памяти с адресами 96 и 97, где хранятся данные для авторизации в пользовательской сети Wi-Fi. Если по этому адресу не было записано значение, то функция возвращает значение 255 и устройство переходит в режим программной точки доступа Wi-Fi. В режиме SoftAP (см. Рис. 15) производится инициализация и запуск сервера. Чтобы обрабатывать входящие HTTP-запросы используется функция on. Эта функция принимает два параметра. Первый − это URL, а второй − имя функции, которая выполняется при нажатии на этот URL. При переходе по корневой ссылке выполняется авторизация пользователя в сети Wi-Fi, по ссылке /ok − происходит обработка данных.

```
server.on("/", handleRoot);
server.on("/ok", handleOk);
server.begin();
```

Рис. 15. Код программы режима SoftAP

Если в EEPROM-памяти есть сохранённые данные, то микроконтроллер переходит в режим клиента и подключается к Wi-Fi сети пользователя с помощью этих данных. В режиме STA (см. Рис. 16) также производится инициализация и запуск сервера. Ссылки /save , /save1, /save2 занимаются обработкой введенных настроек полива, освещения и сброса параметров, которые устанавливает пользователь. Ссылка /configs.json формирует ответ микроконтроллера в формате JSON. Так же происходит обработка перехода по несуществующим ссылкам, с выводом текста «Not Found». Запускаются три подпрограммы tochka(),tochka1(), tochka2() необходимые для отображения точки времени-влажности на графике влажности почвы, а так же для отображения данных последнего времени измерения влажности и ее значения в верхнем блоке веб-страницы.

```
server.onNotFound([]() {
  if (!handleFileRead(server.uri()))
    server.send(404, "text/plain", "Not Found");
});
server.on("/save", handleSave);
server.on("/save1", handleSave1);
server.on("/save2", handleSave1);
server.on("/configs.json", handle_ConfigJSON);
server.begin();

ee  = tochka1();
ee.trim();
hh  = tochka2();
tochka();
```

Рис. 16. Код программы режима STA

В основном цикле (см. Рис. 17) программы для обработки фактических входящих HTTP-запросов вызвана функция handleClient(), функция handleFTP() необходима для обработки запросов от FTP-сервера. Затем в цикле проверки времени интервала измерения влажности вызываются функции отвечающие за работу датчика влажности, работу с временными параметрами для графика влажности почвы и работы освещения.

```
void loop() {
  server.handleClient();
  ftpSrv.handleFTP();

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    ee  = tochka1();
    ee.trim();
    hh  = tochka2();
    get_data_soilmoisture() ;
    tochka();
  }
}
```

Рис. 17. Код основного цикла программы

Полив растений происходит на основании настроек пользователя: с определённой частотой производится замер влажности, если влажность почвы выше минимального порогового значения – растение не поливается, иначе растение поливается в течение 10 секунд. После чего следует измерение влажности, если оно ниже максимального порогового значения, то цикл поливов и замеров повторяется до тех пор, пока значение влажности почвы не станет равным или выше максимального порогового значения.

Освещение включается при достижении времени старта освещения, и выключается при достижении времени окончания освещения, которые сравниваются с текущим временем, полученным через NTP-сервер.

Файл index.html отвечающий за отображение веб-страницы хранится в SPIFFS-памяти устройства, а для удобной записи файлов в данную память был создан FTP-сервер на базе микроконтроллера.

При создании веб-страницы были использованы языки HTML, CSS, Javascript, а так же подключена библиотека jQuery [43], включающая в себя набор функций языка Javascript, фокусирующийся на взаимодействии JavaScript и HTML. Библиотека jQuery предоставляет удобный API для работы с AJAX (Asynchronous Javascript and XML), который в свою очередь обеспечивает обмен данными веб-страницы и веб-браузера без перезагрузки страницы. Для

этого используется технология динамического обращения к серверу с использованием XMLHttpRequest.

XMLHttpRequest (XMLHTTP, XHR) − API, доступный в скриптовых языках браузеров, таких как JavaScript. Использует запросы HTTP или HTTPS напрямую к веб-серверу и загружает данные ответа сервера напрямую в вызывающий скрипт. Информация может передаваться в любом текстовом формате, например, в XML, HTML или JSON [39].

Для отправки и принятия данных от веб-сервера был выбран формат JSON, так как он удобней для интерпретации как стороне веб-страницы, так и на стороне микроконтроллера.

Протокол HTTP использует методы GET и POST для отправки данных на сервер [37]. Метод POST имеет большую степень защиты и параметры запроса не видны пользователю, что дает данному методу преимущество при пересылке конфиденциальных данных, этот метод будет использоваться на странице авторизации пользователя в сети Wi-Fi. При работе со страницей управления устройством используются методы GET.

Для создания диаграммы влажности почвы были использованы бесплатно распространяемые инструменты Google chart [49], позволяющие размещать на веб-странице различные диаграммы. Эти диаграммы основаны на технологии HTML5 / SVG. Они являются интерактивными, и многие из них можно переносить и масштабировать. Данные для диаграммы формируются микроконтроллером.

Управление устройством по уходу за растениями возможно как с персонального компьютера, так и с мобильных устройств. Существует два основных подхода для создания сайтов, легко адаптирующихся для разных типов устройств [27]:

- Responsive Design (RWD) − отзывчивый дизайн − проектирование сайта с определенными значениями свойств, например, гибкая сетка макета, которые позволяют одному макету работать на разных устройствах;

- Adaptive Design (AWD) − адаптивный дизайн, или динамический показ − проектирование сайта с условиями, которые изменяются в зависимости от устройства, базируясь на нескольких макетах фиксированной ширины.

При создании веб-страницы предпочтение отдалось отзывчивому дизайну, так как создание нескольких макетов для страницы управления устройством избыточно и занимает много памяти устройства. После создания была пройдена проверка на адаптивность страницы при помощи сервиса Google Mobile Friendly [29] (см. Рис. 18).

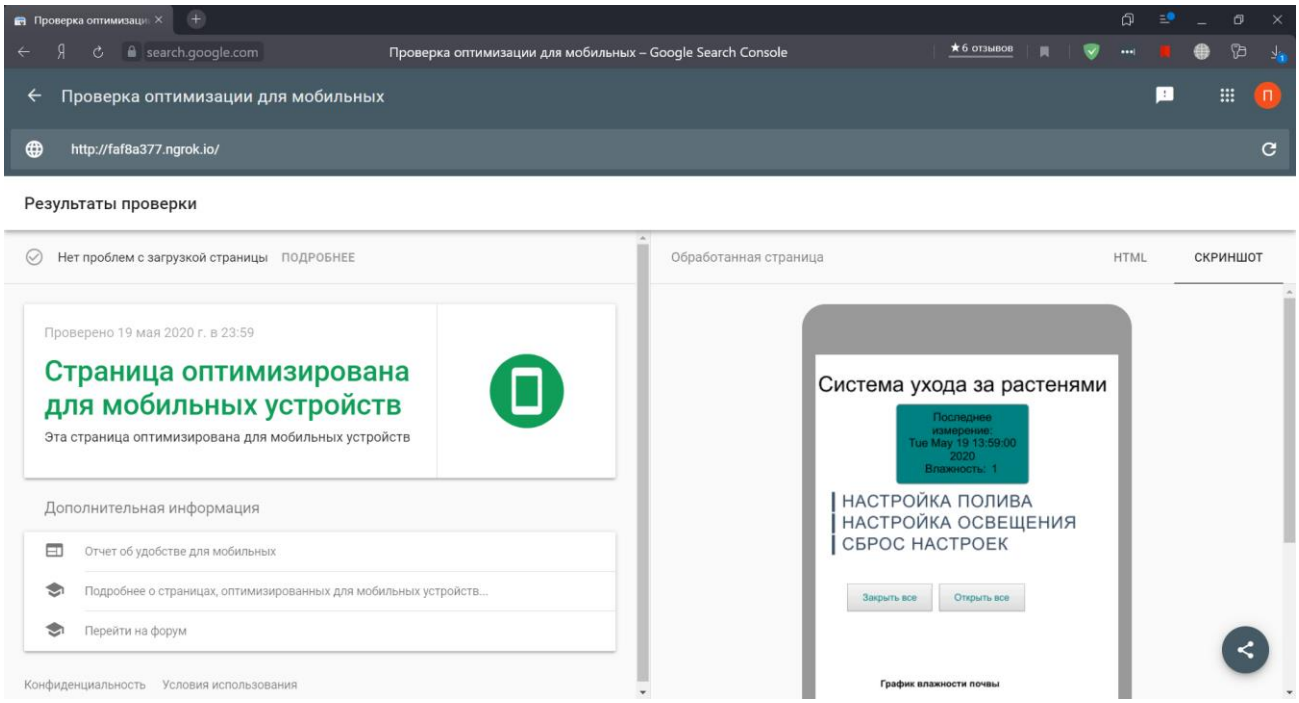

Рис. 18. Проверка адаптивности сайта сервисом Google Mobile Friendly

Работа микроконтроллера может осуществляться в трех разных режимах Wi-Fi:

- станция (STA). В этом режиме ESP-12E получает IP-адрес от беспроводного маршрутизатора, к которому он подключен. С этим IP-адресом он может настроить веб-сервер и доставлять веб-страницы на все подключенные устройства в существующей сети Wi-Fi;
- программная точка доступа (softAP). В этом режиме микроконтроллер создает свою собственную WiFi сеть и действует как концентратор (точно так же как маршрутизатор WiFi) для одной или нескольких станций. В

отличие от WiFi-роутера, он не имеет подключения к проводной сети. Максимальное количество станций, которые могут подключиться к нему, ограничено пятью. В режиме точка доступа создает новую WiFi сеть и устанавливает SSID (имя сети) и IP-адрес для нее. С помощью этого IP-адреса он может доставлять веб-страницы на все подключенные устройства в своей собственной сети;

- комбинированный − станция + программная точка доступа (STA+softAP).

Для подключения устройства к существующей домашней Wi-Fi сети необходимо создать веб-интерфейс для авторизации в сети. Для этого микроконтроллер при первом запуске работает в режиме softAP и создает свою точку доступа. После подключения к Wi-Fi сети «Care_device» (см. Рис. 19) необходимо открыть веб-интерфейс по адресу 192.168.4.1, где располагается интерфейс авторизации в сети (см. Рис. 20).

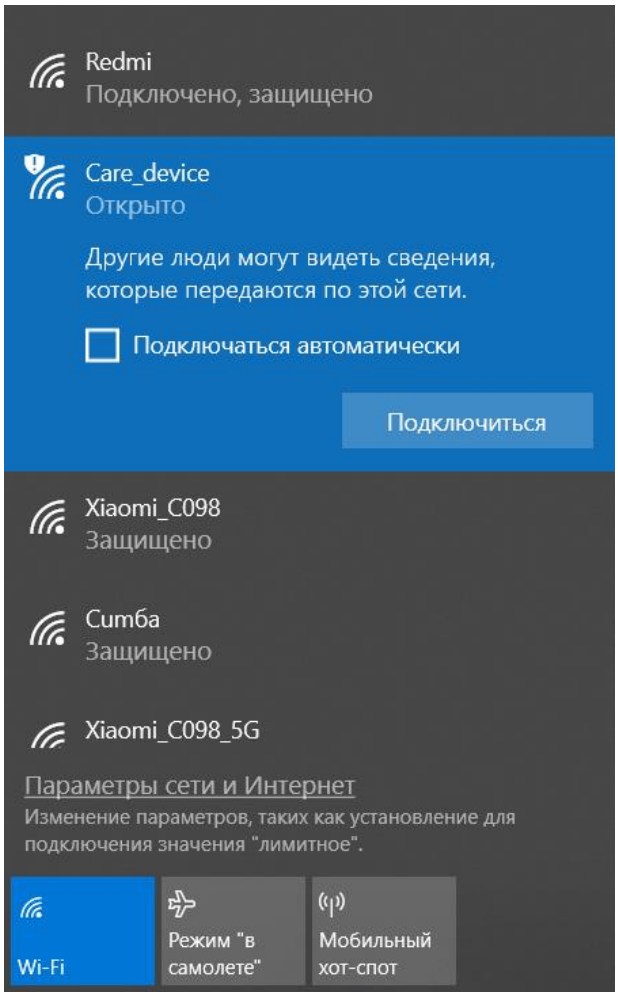

Рис. 19. Выбор Wi-Fi сети «Care_device»

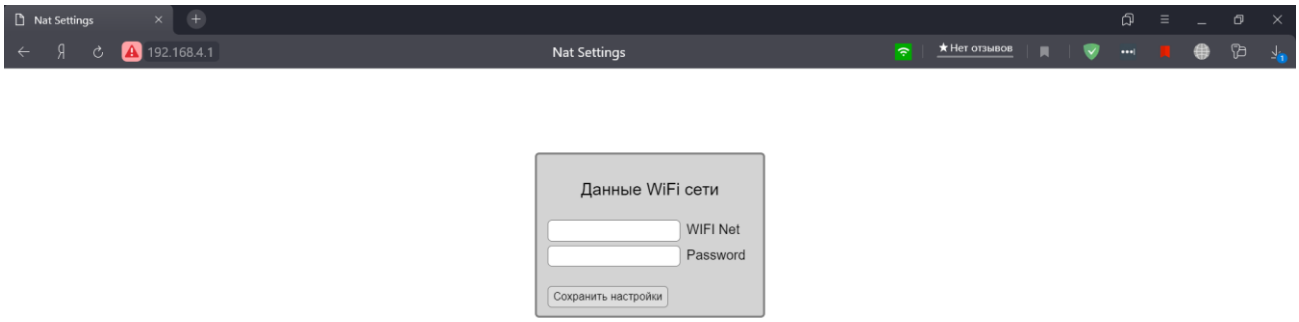

Рис. 20. Веб-интерфейс авторизации в сети

После правильного ввода названия и пароля от домашней Wi-Fi сети пользователь увидит сообщение «Ваши данные сохранены в памяти устройства. Изменения применятся после перезагрузки устройства». Если данные введены неверно и Wi-Fi сеть не была найдена, выведется сообщение «Не найдена WIFI сеть». В этом случае необходимо ввести данные заново.

После перезагрузки устройства данные сети будут записаны в EEPROM-память. При последующих включениях устройства эти данные будут использоваться при подключении к данной Wi-Fi сети без необходимости повторной авторизации.

Веб-интерфейс управления устройством располагается по адресу 192.168.31.33. Пользователю доступны три раздела настроек:

- настройка полива (см. Рис. 21) с возможностью настройки частоты измерения влажности и пороговых значений. Частота измерений указывается в часах. Максимальное пороговое значение отвечает за верхнюю границу влажности почвы, при которой полив растения заканчивается. Минимальное пороговое значение отвечает за нижнюю

границу влажности почвы, при которой устройство начинает полив растения. Значения указываются в процентном значении, где 0 % − почва сухая, 100 % − почва сырая;

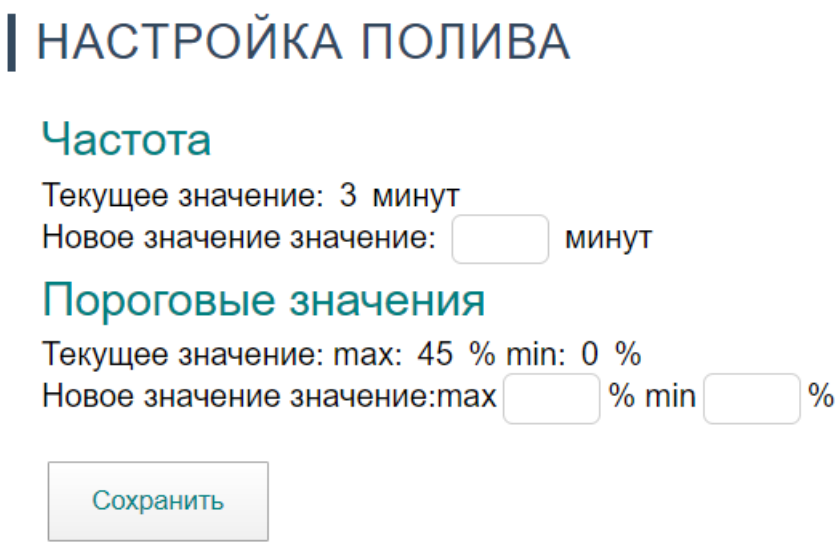

Рис. 21. Веб-интерфейс настройки полива

- настройка освещения (см. Рис. 22) дает пользователю возможность настроить время включения и выключения освещения растения. Время задается в формате ЧЧ:ММ с отображением в 24-часовом формате;

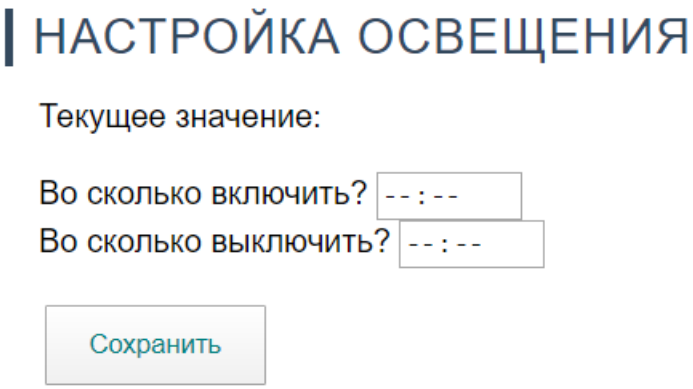

Рис. 22. Веб-интерфейс настройки освещения

- сброс настроек (см. Рис. 23) позволяет пользователю сбросить устройство в начальное состояние, в том числе данные для авторизации пользователя в сети Wi-Fi удаляются из памяти устройства, и потребуется повторная авторизация.

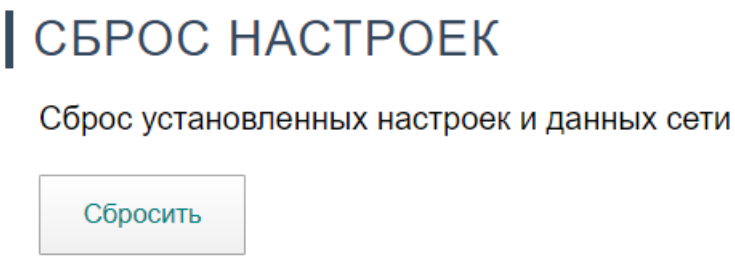

Рис. 23. Веб-интерфейс сброса настроек

2.3 Руководство пользователя

Составлен на основе ГОСТ «19.505-79 ЕСПД. Руководство оператора. Требования к содержанию и оформлению» [14]

1. Введение

1.1. Область применения

Продукт предназначен для домашнего использования.

1.2. Краткое описание возможностей

Автоматизированная система предоставляет следующие возможности:

- Управление системой по уходу за растениями посредством веб-интерфейса;
- Полив растений согласно заданным параметрам и показаниям датчика влажности почвы;
- Освещение растения в указанное пользователем время.

1.3. Уровень подготовки пользователя

Для управления системой пользователь должен иметь минимальный набор знаний и навыков работы с компьютером, браузером и электронными девайсами.

1.4. Перечень эксплуатационной документации

Настоящее «Руководство пользователя»

2. Назначение и условия применения

2.1. Назначение системы

Система предназначена для управления автоматизированной системой по уходу за растениями через веб-интерфейс, обеспечением растений необходимыми условиями, а именно водой и светом.

2.2. Условия применения

Необходимо иметь роутер с подключенным интернетом и устройство с возможностью подключения к домашней Wi-Fi сети. В качестве устройства управления может выступать любой компьютер или ноутбук, с установленным браузером или мобильное устройство на операционной системы Android версии 4.2 и выше или IOS версии 10.3.3 и выше.

3. Подготовка к работе

3.1. Состав и содержание дистрибутивного носителя данных

Система не требует установки, управление происходит через любой веб-браузер.

3.2. Порядок загрузки данных и проверка работоспособности

Перед запуском веб-интерфейса для управления необходимо подключить устройство и разместить его рядом с растением:

1) датчик влажности разместить в горшке с растением так, чтобы контактные ножки были полностью утоплены в почве;
2) разместить систему подачи воды: трубку расположить в горшке с растением, желательно с другой стороны от датчика влажности, если необходимо − воспользоваться проволочным удерживателем трубки. Насос опустить в емкость с водой;
3) разместить светодиодную ленту для освещения растений по своему усмотрению.

Порядок проверки работоспособности:

1) подать питание на устройство. Подключится к Wi-Fi сети «Care_device» и перейти по адресу 192.168.4.1 для прохождения авторизации своей сети. Ввести название сети и пароль. Если подключение произошло – выведется соответствующее сообщение.

Если сеть с таким названием или паролем не будет найдена − необходимо повторно ввести данные. Для сохранения данных в памяти устройства необходима его перезагрузка;

2) подключится к своей домашней сети и перейти по адресу 192.168.31.33. Если откроется веб-интерфейс системы управления устройством – вы подключены к системе.

4. Описание операций

Для пользователя доступны следующие операции:

1) авторизация в домашней Wi-Fi сети;
2) просмотр данных о влажности почвы в виде отдельной информации о последнем измерении и влажности, а так же в виде графика изменения влажности;
3) изменение параметров полива растений: частоты измерения влажности, пороговых минимальных и максимальных значений процентов влажности почвы, при которых начинается и прекращается полив растений;
4) изменение времени включения и выключения освещения;
5) сброс установленных настроек, в том числе и данных для авторизации в сети.

5. АВАРИЙНЫЕ СИТУАЦИИ

При авторизации в домашней сети Wi-Fi выходит сообщение «Не найдена WIFI сеть» − необходимо проверить вводимые данные: название сети и пароля, и ввести их снова.

Если веб-страница управления системой по уходу за растениями не загружается необходимо:

1) перезагрузить устройство;
2) проверить работоспособность роутера и наличие интернета;
3) если проблема не исчезает − провести авторизацию в домашней сети заново.

При неверном вводе данных в поля настройки поливом, пользователю придет сообщение о некорректности данных.

При отключении от домашней Wi-Fi сети устройство продолжит свою работу с сохраненными настройками. При повторном подключении к той же сети авторизация пройдет автоматически. Если подключение произошло к другой сети – необходима повторная авторизация.

6. РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ

Для успешного освоения системы изучить настоящее «Руководство пользователя» и выполнить действия, указанные в пункте 4.

2.4 Результаты апробации

По окончании разработки и после согласования с научным руководителем Арбузовым С.С. было принято решение провести апробацию автоматизированной системы для ухода за растениями в формате выращивания растения с применением данной системы. В качестве растения был выбран листовой салат сорта Афицион. Родиной этого салата считаются страны Средиземноморья. А значит, салат является тепло- и светолюбивый, нуждающийся в обильном поливе, но при этом влажность почвы не должна переходить в переувлажнение, иначе растения начнут гнить.

Для него были подобраны настройки устройства со следующими подходящими ему параметрами: частота измерения – 12 ч. Пороговые значения – 20% и 90 %. Выращивание растения производилось в весенний период с достаточным количеством освещения, поэтому дополнительное освещение не использовалось.

Производилась фотофиксация растения каждые 3-4 дня для наглядного отображения его роста и развития, при использовании автоматизированной системы для ухода за растениями (Рис. 24, Рис. 25, Рис. 26, Рис. 27).

Рис. 24. Первый день

Рис. 25. Третий день

Рис. 26. Седьмой день

Рис. 27. Десятый день

Материалы работы также прошли апробацию в формате публикации.

Арбузов С.С., Чернявская Н.В. Применение микроконтроллеров для обучения программированию автоматизированных систем. // Актуальные вопросы преподавания математики, информатики и информационных технологий [Электронный ресурс] : межвузовский сборник научных работ / Урал. гос. пед. ун-т ; науч. ред. Л. В. Сардак. – Электрон. дан. – Екатеринбург : [б. и.], 2020. – 1 электрон. опт. диск (CD-ROM).

Заключение

При выполнении данной выпускной работы была спроектирована и реализована автоматизированная система для ухода за растениями, предназначенная для освещения и полива растений под управлением пользователя, посредством веб-интерфейса. Для достижения поставленной цели в работе решены следующие задачи:

- Произведён анализ состояния проблемы и подходов к ее решению.
- Произведён анализ и обоснован выбор технологий реализации и необходимых программных платформ.
- Подготовлено формализованное техническое задание, в соответствии с которым произвести разработку комплекса.
- Подготовлена техническая и сопроводительная документация.
- Произведена апробация

Таким образом, следует считать, что результаты разработки соответствуют всем требованиям технического задания, поставленная цель достигнута. Работа носит законченный характер.

Список информационных источников

1. Аникейчик Н.Д., Кинжагулов И.Ю., Федоров А.В. Планирование и управление НИР и ОКР : учебное пособие. – СПб. : Университет ИТМО, 2016. 192 с.
2. Анисимов В.В. Проектирование информационных систем. Часть 1. Структурный подход : конспект лекций. Хабаровск : ДВГУПС, 2005. 112 с.
3. Анисимов В.В., Долгов В.А. Проектирование информационных систем. Часть 2. Объектно-ориентированный подход : конспект лекций. Хабаровск : ДВГУПС, 2007. 112 с.
4. Веб-сервисы // GitHub Gist URL: https://gist.github.com/vchernogorov/81da656048875132d6963304d449f770 (дата обращения: 17.04.2019).
5. Выращивание салата в домашних условиях // AGROSETKA74 URL: https://agrosetka74.ru/ogorod/salat-listovoj-afitsion-kak-vyrashhivat-v-domashnih-usloviyah.html (дата обращения: 03.05.2019).
6. ГОСТ Р 52003-2003. Уровни разукрупнения радиоэлектронных средств. Введ. 2003-07-01. М. : Стандартинформ. 11 с. (Государственный стандарт Российской Федерации. Термины и определения).
7. ГОСТ Р 53791-2010. Стадии жизненного цикла изделий производственно-технического назначения. Введ. 2011-01-01. М. : Стандартинформ. 12 с. (Национальный стандарт Российской Федерации. Ресурсосбережение. Общие положения).
8. ГОСТ 2.103-2013. Стадии разработки. Взамен ГОСТ 2.103-68; Введ. 2015-07-01. М. : Стандартинформ. 8 с. (Межгос. стандарт. Единая система конструкторской документации).
9. ГОСТ 2.105-95. Общие требования к текстовым документам. Взамен ГОСТ 2.105-79, ГОСТ 2.906-71; Введ. 1996-07-01. М. : Всероссийский научно-исследовательский институт стандартизации и сертификации в

машиностроении (ВНИИНМАШ). 27 с. (Межгос. стандарт. Единая система конструкторской документации).

10. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Взамен ГОСТ 7.0-84; Введ. 2004-07-01. Межгос. совет по стандартизации, метрологии и сертификации. М. : Изд-во стандартов. 111 с. (Межгос. стандарт. Система стандартов по информации, библиотечному и издательскому делу. Общие требования и правила составления).
11. ГОСТ 15.101-98. Порядок выполнения научно-исследовательских работ. Взамен ГОСТ 15.101-80; Введ. 2000-07-01. М.: Стандартинформ. 11 с. (Межгос. стандарт. Система разработки и постановки продукции на производство).
12. ГОСТ 19.502-78. Описание применения. Введ. 1980-01-01. М. : Стандартинформ. 2 с. **(**Межгос. стандарт. Единая система конструкторской документации. Требования к содержанию и оформлению).
13. ГОСТ 19.503-79. Руководство системного программиста. Введ. 1980-01-01. М. : Стандартинформ. 4 с. (Межгос. стандарт.. Единая система конструкторской документации. Требования к содержанию и оформлению).
14. ГОСТ 19.505-79. Руководство оператора. Введ. 1980-01-01. М. : Стандартинформ. 2 с. (Межгос. стандарт.. Единая система программной документации. Требования к содержанию и оформлению).
15. ГОСТ 34.602-89. Информационная технология. Взамен ГОСТ 24.201-85; Введ. 1990-01-01. М. : Стандартинформ. 12 с. (Межгос. стандарт. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы).
16. Дополнительное освещение в квартире: лампа для комнатных цветов своими руками // Вдоме club URL: https://vdome.club/obstanovka/poleznoe/lampa-dlya-tsvetov.html (дата обращения: 18.03.2019).

17. И М.Х., Дун Ф.В. Создаем Web-сервисы RESTful при помощи Spring 3. URL: http://www.ibm.com/developerworks/ru/library/waspring3webserv/index.html (дата обращения: 18.03.2019).
18. Каленкович Н.И. Радиоэлектронная аппаратура и основы её конструкторского проектирования : учебно-методическое пособие для студентов спец. «Моделирование и компьютерное проектирование» и «Проектирование и производство РЭС». Минск : БГУИР, 2008. 200 с.
19. Коцюба И.Ю., Чунаев А.В., Шиков А.Н. Основы проектирования информационных систем : учебное пособие. СПб. : Университет ИТМО, 2015. 206 с.
20. Кротова Е.И. Основы конструирования и технологии производства РЭС : учебное пособие. Ярославль : ЯрГУ, 2013. 192 с.
21. Методология IDEF0 // Itteach.ru URL: https://itteach.ru/bpwin/metodologiya-idef0 (дата обращения: 18.03.2019).
22. Микроконтроллер: определение, задачи, разновидности, применение // Future2day URL: https://future2day.ru/mikrokontroller/ (дата обращения: 18.04.2020).
23. Микроконтроллер ESP32 и проекты Arduino // Arduinomaster URL: https://arduinomaster.ru/platy-arduino/esp32-arduino-raspinovka-arduino-ide/ (дата обращения: 21.02.2020).
24. Муминов А.А. Компоненты электронной техники : учебное пособие. Таш ГТУ, 2011. 107с.
25. Настройка программного обеспечения ESP8266 в Arduino IDE // Arduino + URL: https://arduinoplus.ru/programmnoe-obespechenie-esp8266-na-arduino-ide/ (дата обращения: 17.04.2019).
26. Никольский А.П. JavaScript на примерах. СПб. : Наука и Техника, 2017. 272 с.

27. Отзывчивый и адаптивный дизайн сайта // HTML5BOOK URL: https://html5book.ru/otzyvchivyj-dizayn-saita/#part5 (дата обращения: 17.04.2019).
28. Оформитель библиографических ссылок // SNOSKA.INFO URL: http://snoskainfo.ru/ (дата обращения: 17.04.2019).
29. Проверка оптимизации для мобильных // Search Google URL: https://search.google.com/test/mobile-friendly (дата обращения: 17.04.2019).
30. Проектирование информационных систем: учебник и практикум для СПО / под общ. ред. Чистова Д.В. М. : Юрайт, 2019. 258с.
31. Проект Unofficial Development Kit for Espressif ESP8266 // Programs74.ru URL: https://programs74.ru/udkew.html (дата обращения: 17.04.2019).
32. Различия REST и SOAP // Хабр URL: https://habr.com/ru/post/483204/ (дата обращения: 17.04.2019).
33. Рельсы веб-интеграции. REST и SOAP // Intervolga URL: https://www.intervolga.ru/blog/projects/relsy-veb-integratsii-rest-i-soap/ (дата обращения: 17.04.2019).
34. Салат листовой афицион как выращивать в домашних условиях // Dacha-posadka.ru URL: http://dacha-posadka.ru/doma/salat-listovoy-aficion-kak-vyraschivat-v-domashnih-usloviyah.html (дата обращения: 03.05.2019).
35. Семейство микроконтроллеров MSP430x2xx. Архитектура, программирование, разработка приложений . М. : Додэка-XXI, 2010. 544 с.
36. Создание простого веб-сервера NodeMCU ESP8266 в Arduino IDE // Joyta.ru URL: http://www.joyta.ru/12628-sozdanie-prostogo-veb-servera-nodemcu-esp8266-v-arduino-ide/ (дата обращения: 4.04.2020).
37. Ташков П. А. Веб-мастеринг на 100 %: HTML, CSS, JavaScript, PHP,CMS, AJAX, раскрутка. СПб. : Питер, 2010. 512с.
38. Федеральный закон Российской Федерации «Об информации, информационных технологиях и о защите информации» от 08.07.2006 №

149-ФЗ // Российская газета. 2006 г. № 165. с изм. и допол. в ред. от 25.11.2017

39. Флэнаган Д. JavaScrirt: карманный справочник. 3 изд. М. : «ООО И.Д. Вильямс» , 2013. 320 с.
40. Хофманн М. Микроконтроллеры для начинающих: Пер. с нем. СПб. : БХВ-Петербург, 2014. 304с.
41. Черняк Л. SOAP и REST, вместе или порознь? // Открытые системы. 2003. №9. URL: https://www.osp.ru/os/2003/09/183376 (дата обращения: 17.04.2019).
42. Чуб В.В., Лезина К.Д. Все о комнатных растениях. М. : Эксмо, 2002. 336 с.
43. Шевчук А. JQuery учебник для начинающих URL: https://anton.shevchuk.name/jquery-book/ (дата обращения: 17.04.2019).
44. Юхимчук Д. Комнатное цветоводство. Киев : Урожай, 1977. 42 с.
45. Blanchon B. Mastering ArduinoJson: Efficient JSON serialization for embedded C++, 2017. 242 с. URL: https://arduinojson.org/book/serialization_tutorial5.pdf#page=10 (дата обращения: 03.05.2019).
46. Digital 2020: 3.8 Billion People Use Social Media // We are social URL: https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media (дата обращения: 21.02.2020).
47. Espressif smart connectivity platform: ESP8266 URL: https://static.chipdip.ru/lib/140/DOC001140960.pdf (дата обращения: 30.04.2020).
48. Filesystem // ESP8266 Arduino Core URL: https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html (дата обращения: 17.04.2019).
49. Google Chart // Google Developers URL: https://developers.google.com/chart?hl=ru (дата обращения: 17.04.2019).
50. Representational State Transfer (REST) // Techopedia URL: https://www.techopedia.com/definition/1312/representational-state-transfer-rest (дата обращения: 17.04.2019).

51. REST vs SOAP. Часть 1. Почувствуйте разницу // Хабр URL: https://habr.com/ru/post/131343/ (дата обращения: 17.04.2019).
52. SDKs & Demos // Espressif URL: https://www.espressif.com/en/support/download/sdks-demos (дата обращения: 17.04.2019).
53. The MIT License (MIT) // The MIT License (MIT) URL: https://mit-license.org/ (дата обращения: 17.04.2019).
54. What is PlatformIO? // Platformio.org URL: https://docs.platformio.org/en/latest/what-is-platformio.html (дата обращения: 21.02.2020).
55. WiFi ESP8266 в проектах Arduino // Arduinomaster URL: https://arduinomaster.ru/platy-arduino/arduino-esp8266/ (дата обращения: 21.02.2020).

Приложения

Приложение 1

Листинг программы микроконтроллера

```
#include <ESP8266WiFi.h> //Copyright (c) 2015 ekstrand
#include <ESP8266WebServer.h>
#include <EEPROM.h>
#include <time.h>
#include <FS.h>      // Библиотека для работы с файловой системой
#include <ESP8266FtpServer.h> // Copyright (C) 2016 nailbuster
FtpServer ftpSrv;            // Создаём объект для работы с FTP

boolean conf = false;
String e;
int hh;
String ee;

const char* deviceName = "poliv";
// значение полного полив
#define MIN 100
// значение критической сухости
#define MAX 700

int sensor_pin = A0;
int m = 0;
int minr = 0, maxr = 100 ;
const uint16_t lengt = 100; //max kol tochek trenda
uint16_t tick = 0;
time_t tnow[lengt];
float h[lengt];

String date_toch;
String h_toch;
long interval = 100000; //interval zapisi danih dlya trenda
```

```
String f , minread = "0", maxread = "100",ts,tf;
unsigned long previousMillis = 0;
String html_header = "<html>\
 <meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">\
 <head>\
  <title>ESP8266 Settings</title>\
  <style>\
   .bloks_avt{ background-color: #D3D3D3; margin:0 auto; border: 2px solid grey; border-radius: 5px; padding: 10px; width: 250px;}\
    body { font-family: Sans-Serif; }\
  #text { text-align: center; font-size: 20px;}\
  #text1{ font-family: sans-serif; font-size: 15px; display: inline;} \
   input {font-size: 13px; margin: 5px 3px 0 3px; border: 1px solid grey; border-radius: 5px; height:25px;}\
  </style>\
 </head>";

ESP8266WebServer server(80);

void setup(void)
{
 configTime(5 * 3600, 0, "pool.ntp.org"); //NTP sayti dlia schitivaniya vremeni
 delay(500);
 // preparing GPIOs
 pinMode(sensor_pin, OUTPUT);
 digitalWrite(sensor_pin, LOW);

 f = interval / 60000;
byte len_ssid, len_pass;
 delay(3000);
 Serial.begin(115200);
 Serial.println();
 SPIFFS.begin();
 ftpSrv.begin("esp8266", "esp8266");   // Инициализируем FTP-сервер (на 21-й порт)
```

```
EEPROM.begin(98);
/* EEPROM.write(96,0);
  EEPROM.write(97,0);
  EEPROM.commit();
  EEPROM.end();*/
len_ssid = EEPROM.read(96);
len_pass = EEPROM.read(97);
if (len_pass > 64) len_pass = 0;
pinMode(4, INPUT_PULLUP);
pinMode(0, INPUT_PULLUP);
if ((len_ssid < 33) && (len_ssid != 0)) {

 // Режим STATION
 WiFi.mode(WIFI_STA);
 unsigned char* buf_ssid = new unsigned char[32];
 unsigned char* buf_pass = new unsigned char[64];
 for (byte i = 0; i < len_ssid; i++) buf_ssid[i] = char(EEPROM.read(i));
 buf_ssid[len_ssid] = '\x0';
 const char *ssid = (const char*)buf_ssid;
 for (byte i = 0; i < len_pass; i++) buf_pass[i] = char(EEPROM.read(i + 32));
 const char *pass = (const char*)buf_pass;
 buf_pass[len_pass] = '\x0';
 delay(2000);
 Serial.print("SSID: ");
 Serial.print(ssid);
 Serial.print("  ");
 Serial.print("Password: ");
 Serial.println(pass);
 /*
  WiFi.disconnect(); //Prevent connecting to wifi based on previous configuration

  WiFi.hostname(deviceName); // DHCP Hostname
  WiFi.config(ip, subnet, gateway, dns);*/
 WiFi.begin(ssid, pass);
```

```
// Wait for connection
while ( WiFi.status() != WL_CONNECTED ) {
 delay (500 );
 Serial.print ( "." );
}
Serial.println();
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

server.onNotFound([]() {                          // Описываем действия при событии "Не найдено"
 if (!handleFileRead(server.uri()))                // Если функция handleFileRead (описана ниже) возвращает значение false в ответ на поиск файла в файловой системе
  server.send(404, "text/plain", "Not Found");     // возвращаем на запрос текстовое сообщение "File isn't found" с кодом 404 (не найдено)
});
server.on("/save", handleSave);
server.on("/save1", handleSave1);
server.on("/save2", handleSave1);
server.on("/configs.json", handle_ConfigJSON);  // формирование configs.json страницы для передачи данных в web интерфейс
SPIFFS.begin();                 // Инициализируем работу с файловой системой
server.begin();
tochka();

}
else // Режим SoftAP
{
 const char *ssid_ap = "Care_device";
 WiFi.mode(WIFI_AP);
 Serial.print("Configuring access point...");
 /* You can remove the password parameter if you want the AP to be open. */
```

```
 WiFi.softAP(ssid_ap);
 delay(2000);
 Serial.println("done");
 IPAddress myIP = WiFi.softAPIP();
 Serial.print("AP IP address: ");
 Serial.println(myIP);

 server.on("/", handleRoot);
 server.on("/ok", handleOk);
 server.begin();
 SPIFFS.begin();         // Инициализируем работу с файловой системой
 Serial.println("HTTP server started");

 }

}
void loop() {
 server.handleClient();
 ftpSrv.handleFTP();
 // Перевод модуля в режим конфигурации путем замыкания GPIO5 на массу

 if ((digitalRead(4) == LOW) && !conf) {
 EEPROM.write(96, 255);
 EEPROM.commit();
 EEPROM.end();
 conf = true;
 Serial.println("Please reboot module for coniguration --gpio5--");
 Serial.println(conf);
 // Serial.println(f_state);
 }

 unsigned long currentMillis = millis();
 if (currentMillis - previousMillis >= interval) {
 previousMillis = currentMillis;
```

```
 ee = tochka1();
  ee.trim();
  hh = tochka2();
  get_data_soilmoisture() ;
  tochka();

 }

}



void tochka() {
 h[tick] = get_data_soilmoisture();
 tnow[tick] = time(nullptr);
 Serial.print("Дата: " + String(ctime(&tnow[tick])));
 Serial.print("Проценты: ");
 Serial.print(h[tick]);
 if (tick < lengt - 1) tick ++; else tick = 0;
}
/*
void tchk(){
  uint16_t k, y=0;
  for (int i=1; i <= lengt; i++){
   k = tick-1 + i;
   if (h[k]>0){
    if (y>0) web += ",";
    y ++;
    if (k>lengt-1) k = k - lengt;

    String(tnow[k]-(5*3600)); //2- chasovoy poyas
   //*1000),
   h[k];
```

```
   }
  }

}
*/
String tochka1() {
 tnow[tick] = time(nullptr);
 e = ctime(&tnow[tick]);

 return e;
}

int tochka2() {
 h[tick] = get_data_soilmoisture();
 int v = h[tick];
 return v;
}

float get_data_soilmoisture() {
 int avalue = analogRead(sensor_pin);
 Serial.println();
 Serial.print("С датчика: ");
 Serial.println(sensor_pin);
 //масштабируем значение в проценты
 avalue = constrain(avalue, MIN, MAX);
 int moisture = map(avalue, MIN, MAX, 100, 0);
 return (float)moisture;
}

int inProcent(int zna) {
 zna = map(zna, MIN, MAX, 0, 100);
 return zna;
}
```

```
int inSoprot(int zna) {
 zna = map(zna, 0, 100, MIN, MAX);
 return zna;
}

void handleSave() {
 if (server.arg("fr") != "") {
  f = server.arg("fr");
  // ++++++++++++Парсинг частоты++++++++++++++++++++
  String MyStr = f;
  char floatbufVar[32];
  MyStr.toCharArray(floatbufVar, sizeof(floatbufVar));
  float f = atof(floatbufVar);
  //interval =f*3600000; //из часов в мс
  if (f != 0) {
   interval = f * 60000; //из минут в мс
   Serial.print("Интервал f: ");
   Serial.print(interval);
  }
 }
 if (server.arg("minr") != "") {
  minread = server.arg("minr");
  int minr1 = inSoprot(minread.toInt());
  Serial.println("minr: " + minread);
  Serial.print("inSoprot: ");
  Serial.print(minr1);
 }
 if (server.arg("maxr") != "") {
  maxread = server.arg("maxr");
 }

 /* Serial.println("fr: " + f);
  Serial.println("minr: " + minread);
  Serial.println("maxr: " + maxread);*/
```

```
 server.send(200, "text/plane", "OK");
}
void handleSave1() {
 if (server.arg("time_start") != "") {
  ts = server.arg("time_start");
  Serial.println("time_start: " + ts);
 }
 if (server.arg("time_fin") != "") {
  tf = server.arg("time_fin");
  Serial.println("time_fin: " + tf);
 server.send(200, "text/plane", "OK");
}
}
void handle_ConfigJSON() {
 String json = "{"; // Формировать строку для отправки в браузер json формат
 //{"SSDP":"SSDP-
test","ssid":"home","password":"i12345678","ssidAP":"WiFi","passwordAP":"","ip":"192.168.0.10
1"}
 // Частота
 json += "\"fr\":\"";
 json += f;
 // Минималь
 json += "\",\"minr\":\"";
 json += minread;
 // Максимал
 json += "\",\"maxr\":\"";
 json += maxread;
  json += "\",\"time_iz\":\"";
 json += ee;
  json += "\",\"hum_iz\":\"";
 json += hh;
 json += "\"}";
 server.send(200, "text/json", json);
}
```

```
void handleRoot() {
 String str = "";
 str += html_header;
 str += "<body>\
  <form method=\"POST\" action=\"ok\">\
  <div class = \"bloks_avt\">\
  <p id=\"text\"> Данные WiFi сети </p>\
   <input name=\"ssid\"> WIFI Net</br>\
   <input name=\"pswd\"> Password</br></br>\
   <input type=SUBMIT value=\"Сохранить настройки\">\
   </div>\
  </form>\
 </body>\
</html>";
 server.send ( 200, "text/html", str );
}

void handleOk() {
 String ssid_ap;
 String pass_ap;
 unsigned char* buf = new unsigned char[64];
 String str = "";
 str += html_header;
 str += "<body>";

 EEPROM.begin(98);

 ssid_ap = server.arg(0);
 pass_ap = server.arg(1);

 if (ssid_ap != "") {
  EEPROM.write(96, ssid_ap.length());
  EEPROM.write(97, pass_ap.length());
  ssid_ap.getBytes(buf, ssid_ap.length() + 1);
```

```
 for (byte i = 0; i < ssid_ap.length(); i++)
  EEPROM.write(i, buf[i]);

 pass_ap.getBytes(buf, pass_ap.length() + 1);
 for (byte i = 0; i < pass_ap.length(); i++)
  EEPROM.write(i + 32, buf[i]);
 EEPROM.commit();
 EEPROM.end();
 str += " <div class = \"bloks_avt\">\
 <p id=\"text1\"> Ваши данные сохранены в памяти устройства</br>\
 Изменения применятся после перезагрузки устройства</br></br> \
  <a href=\"/\">Return</a> to settings page</p></br>\
  </div>";
}
 else {
  str += " <div class = \"bloks_avt\">\
  <p id=\"text1\"> Не найдена WIFI сеть</br>\
  <a href=\"/\">Перезагрузить</a> стараницу настроек</br></p>\
    </div>";
 }
 str += "</body></html>";
 server.send ( 200, "text/html", str );
}

bool handleFileRead(String path) {                // Функция работы с файловой системой
 if (path.endsWith("/")) path += "index.html";          // Если устройство вызывается по корневому адресу, то должен вызываться файл index.html (добавляем его в конец адреса)
 String contentType = getContentType(path);             // С помощью функции getContentType (описана ниже) определяем по типу файла (в адресе обращения) какой заголовок необходимо возвращать по его вызову
 if (SPIFFS.exists(path)) {                       // Если в файловой системе существует файл по адресу обращения
  File file = SPIFFS.open(path, "r");                 // Открываем файл для чтения
```

size_t sent = server.streamFile(file, contentType); // Выводим содержимое файла по HTTP, указывая заголовок типа содержимого contentType

file.close(); // Закрываем файл

return true; // Завершаем выполнение функции, возвращая результатом ее исполнения true (истина)

}

return false; // Завершаем выполнение функции, возвращая результатом ее исполнения false (если не обработалось предыдущее условие)

}

String getContentType(String filename) { // Функция, возвращающая необходимый заголовок типа содержимого в зависимости от расширения файла

if (filename.endsWith(".html")) return "text/html"; // Если файл заканчивается на ".html", то возвращаем заголовок "text/html" и завершаем выполнение функции

else if (filename.endsWith(".css")) return "text/css"; // Если файл заканчивается на ".css", то возвращаем заголовок "text/css" и завершаем выполнение функции

else if (filename.endsWith(".js")) return "application/javascript"; // Если файл заканчивается на ".js", то возвращаем заголовок "application/javascript" и завершаем выполнение функции

else if (filename.endsWith(".png")) return "image/png"; // Если файл заканчивается на ".png", то возвращаем заголовок "image/png" и завершаем выполнение функции

else if (filename.endsWith(".jpg")) return "image/jpeg"; // Если файл заканчивается на ".jpg", то возвращаем заголовок "image/jpg" и завершаем выполнение функции

else if (filename.endsWith(".gif")) return "image/gif"; // Если файл заканчивается на ".gif", то возвращаем заголовок "image/gif" и завершаем выполнение функции

else if (filename.endsWith(".ico")) return "image/x-icon"; // Если файл заканчивается на ".ico", то возвращаем заголовок "image/x-icon" и завершаем выполнение функции

return "text/plain"; // Если ни один из типов файла не совпал, то считаем что содержимое файла текстовое, отдаем соответствующий заголовок и завершаем выполнение функции

}

Приложение 2

Листинг HTML

```
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <meta charset="utf-8">
    <meta http-equiv="refresh" content="1000"/>

    <title>Уход за растениями</title>
   <script type='text/javascript' src='https://www.gstatic.com/charts/loader.js'></script>
   <script type="text/javascript" src='http://ajax.googleapis.com/ajax/libs/jquery/1.7.0/jquery.min.js'></script>

<script type="text/javascript">
$(document).ready(function(){
 $('.block_link').click(function(){
 $(this).parent().children('div.bloks_set').toggle('normal');
 return false;
 });
});
</script>

<script>
function sendData() {
var fr = document.getElementById('freq').value;
var maxread = document.getElementById('maxread').value;
var minread = document.getElementById('minread').value;

  if (maxread <0 || maxread >100) {
   alert("Ошибка");
   return 0;
  }
```

```
 var xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {

                    }
 };
var data = "fr="+fr + "&minr=" + minread + "&maxr=" + maxread;

 xhttp.open("GET", "save?" + data, true);
 xhttp.send();

 setInterval(function() {
 // Call a function repetatively with 2 Second interval
 getData();
}, 500); //2000mSeconds update rate
 clear();
}

function sendData1() {
var time_start = document.getElementById('time_start').value;
var time_fin = document.getElementById('time_fin').value;

 var xhttp = new XMLHttpRequest();

var data1 = "time_start="+time_start + "&time_fin=" + time_fin ;

 xhttp.open("GET", "save1?" + data1, true);
 xhttp.send();

 clear();
}

 function getData() {
 var xhr = new XMLHttpRequest();
```

```
 xhr.open('GET', '/configs.json');
 xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
 xhr.responseType = 'json';
 xhr.send();
 xhr.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var obj = xhr.response;
          document.getElementById("fr").innerHTML = (obj.fr)
                  document.getElementById("maxim").innerHTML = (obj.maxr)
                  document.getElementById("minim").innerHTML = (obj.minr)
                  document.getElementById("date_izm").innerHTML = (obj.time_iz)
                  document.getElementById("humidity").innerHTML = (obj.hum_iz)

  }
 };

}
 function clear() {
     document.getElementById('freq').value = '';
     document.getElementById('maxread').value = '';
                             document.getElementById('minread').value = '';
                              document.getElementById('time_start').value = '';
                             document.getElementById('time_fin').value = '';
    }
</script>
  <script type='text/javascript'>

           google.charts.load('current', {
       'packages': ['corechart']
      });
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = new google.visualization.DataTable();
        data.addColumn('datetime', 'Time');
```

```
        data.addColumn('number', 'Влажность, %');
                                    data.addRows([[new Date(1589298569 * 1000), 0.00], [new
Date(1589298191 * 1000),25.00]]);
        var options = {
          width: '100%',
          title: 'График влажности почвы',
          legend: {
            position: 'bottom'
          },
          hAxis: {
            format: 'dd.MM.yyyy HH:mm',
            gridlines: {
              count: 10,
            },
          }
        };
        function resize() {
          var chart = new
google.visualization.LineChart(document.getElementById('curve_chart'));
          var formatter = new google.visualization.DateFormat({
            pattern: 'dd.MM.yyyy HH:mm'
          });
          formatter.format(data, 0);
          chart.draw(data, options);
        }
        window.onload = resize();
        window.onresize = resize;
      }

         setInterval(function() {
                drawChart();
                    }, 2000);
    </script>
  <script>
```

```
 getData();

   </script>
  <style>
   * {
    margin: 0;
    padding: 0;
   }

   html {

    font-family: sans-serif;
                                margin: 10px auto;
   }
                        .bloks_set {display:none;}
                        .block_link {cursor:pointer;}

                        button {
    color: #008080;
    padding: 10px 20px;
   float:center;
   }
                #butt{
                color: #008080;
    padding: 10px 20px;
                text-decoration: none;
                        margin: 4px 2px;
cursor: pointer;
                }

   #text1 {
    margin-top: 5px;
    margin-bottom: 5px;
    font-size: 20px;
```

```
}

#text2 {
  margin-top: 5px;
  margin-bottom: 5px;
  font-size: 23px;
  color: #008080;
}
#text {
  font-size: 16px;
  display: inline;
}
input[type=text] {
  font-size: 13px;
  margin: 0 3px 0 3px;
  border: 1px solid #cecece;
  border-radius: 5px;
                                font-family: sans-serif;
                                height: 25px;
}

div {
  margin: 0 3px 0px 3px;
  font-family: sans-serif;
}

.bloks_set {
  margin: 10px auto;

  padding: 5px 0 5px 37px;
}
.bloks_vverx {
  text-align: center;
  margin: 10px auto;
```

```
                                width:40%;
    border: 1px solid grey;
    border-radius: 5px;
    padding: 10px;
    background-color: #008080;
                                display: block;
  }

  h1 {
    text-align: center;
    font-weight: 100;
    font-size: 30px;
    margin: 15px 0 10px 0;
  }
                        #hh{
                         margin: 25px 3px 0 15px;
                         color: #008080;
    padding: 10px 27px;
   float:center;
                        }

a {
 text-decoration: none;
 display: inline-block;
 padding: 5px 10px;
 letter-spacing: 1px;
 margin: 0 20px;
 font-size: 24px;

 transition: .3s ease-in-out;
  color: #FFD201;
 letter-spacing: 1px;
 border-bottom: 1px solid transparent;
 border-top: 1px solid transparent;
```

```
 color:#34495e;
 line-height: 1.2;
 position: relative;
 padding: 0 14px;
 text-transform: uppercase;
}
a:after {
 content: "";
 height: 100%;
 min-width: 4px;
 background: #34495e;
 position: absolute;
 left: 0;
 bottom: 0;
 transition: .5s;
}
a:hover:after {
 min-width: 100%;
 background: #95a5a6;
 opacity: .35;
}
    </style>
    <h1>Система ухода за растенями</h1>
 </head>
 <body>

  <div class="bloks_vverx">
      <p id="text">Последнее измерение:</p>
      <div style="display: inline-block;" id="date_izm"></div>
      <p id="text">Влажность: </p>
      <div style="display: inline-block;" id="humidity"></div>
      <br>
    </div>
```

```
<div >
 <a href="" class="block_link"> Настройка полива</a>
 <div class="bloks_set">

    <p id="text2">Частота </p>
    <p id="text">Текущее значение: </p>
    <div style=" display: inline-block;" id="fr"></div>
    <p id="text">минут</p>
    <br>
    <p id="text">Новое значение значение:</p>
    <input type="text" name="freq" id="freq" size=3>
    <p id="text">минут </p>
    <p id="text2">
     Пороговые значения <br>
    </p>
    <p id="text">Текущее значение: max:</p>
    <div style="display: inline-block; " id="maxim"></div>
    <p id="text">%</p>
    <p id="text">min: </p>
    <div style="display: inline-block;" id="minim"></div>
    <p id="text">%</p>
    <br>
    <p id="text">
     Новое значение значение:max<input type="text" name="maxread" id="maxread"
size=3>%
    </p>
    <p id="text">
     min<input type="text" name="minread" id="minread" size=3>%
    </p>

    <div>
     <br>
     <button type="button" id="save_button" onclick="sendData()">Сохранить</button>
    </div>
```

```
 </div>
</div>

<div>
 <a href="" class="block_link"> Настройка освещения</a>
 <div class="bloks_set">

     <p id="text">Текущее значение:
     <div style="display: inline-block;" id="time_dtart_tek"></div>
</p><br>
<p id="text">
  Во сколько включить? <input type="time" name="time_start" id="time_start" size=3>
</p>
<br>
<p id="text">
  Во сколько выключить? <input type="time" name="time_fin" id="time_fin" size=3>
</p>
<div>
  <br>
  <button id="save_button1" onClick="sendData1()">Сохранить</button>
</div>
 </div>
</div>
  <div >
<a href="" class="block_link">Сброс настроек</a>
 <div class="bloks_set">

  <p id="text">Сброс установленных настроек и данных сети</p>
  <br>
  <br>
  <button id="sbros" onClick="window.location.reload( true );">Сбросить</button>
</div>
</div>
 <div id="hh">
```

```
<input id="butt" type="button" value="Закрыть все" on-
click=$("div[class^='bloks_set']").hide('normal')>
<input id="butt" type="button" value="Открыть все" on-
click=$("div[class^='bloks_set']").show('normal')>
</div>
<div id='curve_chart' style='width: 100%; height: 600px'></div>
 </body>
</html>
```